Министерство образования и науки Российской Федерации
Государственное образовательное учреждение
высшего профессионального образования
«Санкт-Петербургский государственный политехнический университет»
Факультет управления и информационных технологий

Зав. каф. КИТвП, к.т.н.
_____ А.В. Речинский
«__»_____ 20__ г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ
РАЗРАБОТКА СИСТЕМЫ УПРАВЛЕНИЯ УЧЕБНЫМИ МАТЕРИАЛАМИ НА ОСНОВЕ СЕМАНТИЧЕСКИХ МОДЕЛЕЙ ПРЕДМЕТНЫХ ОБЛАСТЕЙ

направление: 230100 «Информатика и вычислительная техника»

Выполнил:
Глазунов Вадим Валерьевич
Подпись _____

Руководитель:
доцент ФПС СПбГПУ
Кетов Дмитрий Владимирович
Подпись _____

Рецензент:
ст. преп. ФПС СПбГПУ
Архимандритов Игорь Борисович
Подпись _____

Консультант:
доцент ФУИТ СПбГПУ, к.т.н.
Кудрявцев Дмитрий Вячеславович
Подпись _____

Санкт-Петербург
2011

Реферат

SEMANTIC WEB, ОНТОЛОГИИ, УЧЕБНЫЕ КУРСЫ, СЕМАНТИКА, ПРЕДМЕТНАЯ ОБЛАСТЬ, ONTOPIA

Работа посвящена проектированию и разработке системы управления материалами учебных курсов на основе онтологической модели предметной области. Актуальность решаемой задачи определяется практическим отсутствием систем такого класса в условиях постоянно возрастающей роли индивидуального и дистанционного обучения при подготовке и повышении квалификации специалистов.

Для решения поставленной задачи построены базовая структура учебных материалов, на основе которой могут строиться модели различных предметных областей, и базовая онтологическая модель учебного курса. Адекватность этих моделей продемонстрирована путем описания одной предметной области (формальная семантика языков программирования) и двух учебных курсов, посвященных изучению различных фрагментов предметной области и отличающихся друг от друга целью и базовым уровнем подготовки слушателей. Автором предложена методика создания курсов на основе семантической модели предметной области и реализована система управления такими учебными курсами.

Практическая ценность данной работы заключается в разработке системы управления курсами, позволяющей преподавателям повторно использовать ранее созданные учебные материалы, упростить подготовку индивидуальных программ и вариантов курса, а для слушателей — предоставляет дополнительные, по сравнению с традиционными системами, возможности для поиска и освоения учебных материалов.

Теоретическая ценность работы состоит в предложенных онтологических моделях и методике создания курсов.

Объем работы — 88 страницы. Текст работы содержит 14 иллюстраций, 3 таблицы и 30 ссылок на источники

СОДЕРЖАНИЕ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

AJAX — Asynchronous Javascript And XML – Асинхронный яваскрипт и XML

API — Application Programming Interface – Интерфейс прикладного программирования

APL — Apache Public License – Публичная лицензия Apache

CIG — Common Interest Group – Группа с общими интересами

CSV — Comma-Separated Values – Разделенные запятой значения

FOAF — Friend Of A Friend – Друг друга

GNU — GNU's Not Unix – GNU – это не UNIX

GPL — General Public License – Универсальная общественная лицензия

HTML — HyperText Markup Language – Язык разметки гипертекста

HTTP — TyperText Transfer Protocol – Протокол передачи гипертекста

ISO — International Organization for Standardization – Международная организация по стандартизации

JSON — JavaScript Object Notation – Нотация объекта JavaScript

JSP — Java Server Pages – Серверные страницы Java

LMS — Learning Management System – Система управления обучением

LCMS — Learning Content Management System – Система управления контентом в обучении

LOM — Learning Object Metadata – Метаданные объекта обучения

OKI — Open Knowledge Initiative – Инициатива открытого знания

OSID — Open Service Interface Definitions – Определения интерфейса открытых служб

OSL — Ontopia Schema Language – Язык схем Ontopia

OWL — Web Ontology Language – Язык веб-онтологии

RDBMS — Relational DataBase Management System – Система управления реляционными базами данных

RDF — Resource Description Framework – Фреймворк описания ресурса

RDFS — RDF Schema – Схема фреймворка описания ресурса (также называется словарями RDF)

SCORM — Sharable Content Object Reference Model – Модель экземпляра объекта с разделяемым содержимым

SPARQL — Simple Protocol and RDF Query Language – Простой протокол и язык запросов RDF

SKOS — Simple Knowledge Organization System – Система простой организации знаний

SQL — Structured Query Language – Язык структурированных запросов

SSL — Secure Sockets Layer – Уровень безопасных сокетов

TMQL — Topic Maps Query Language – Язык запросов Topic Maps

URL — Uniform Resource Locator – Единый указатель ресурса

URI — Uniform Resource Identifier – Единый идентификатор ресурса

WYSIWYG — What You See Is What You Get – Что ты видишь – это то, что ты получишь

XML — eXtensible Markup Language – Расширяемый язык разметки

XTM — XML Topic Maps – XML для Topic Maps

ВВЕДЕНИЕ

Современные методы образования предполагают доступность учебных материалов для самостоятельной работы студентов. Многие университеты предоставляют своим студентам или всем желающим возможность получить в электронной форме конспекты лекций, задачи, экзаменационные вопросы по некоторым или всем читаемым в университете курсам. Тем не менее, в большинстве случаев пользователям предоставляются сильно ограниченные возможности поиска необходимых материалов: редко возможно определить место курса в учебном плане, его структуру, найти необходимые определения или задачи, не прочитав все доступные для этого курса материалы.

Даже при наличии эффективных инструментов поиска, в роли которых могут выступать поисковые системы (Google, Яндекс), одной только доступности материалов недостаточно для организации эффективного обучения. Учебные курсы характеризуются не только своим содержанием, но и логическим порядком следования материала. При создании курса автор выбирает материал и порядок его изложения, исходя из целей курса и возможностей аудитории, на которую курс рассчитан. При публикации учебных материалов для широкого круга читателей достаточно трудно делать предположения о целях, которые преследуют читатели, и об их уровне подготовки. В зависимости от выбранной читателем цели одни и те же материалы могут быть обязательными для изучения или играть роль необязательных дополнений. Читателю, уровень подготовки которого не соответствует тому базовому уровню, на который рассчитывал автор, может быть необходимо предварительно обратиться к другим источникам. Чтобы обучение было эффективным, необходимо помочь читателю сориентироваться в предлагаемых ему материалах и смежных вопросах. Примером традиционного подхода к ориентации читателя являются предисловия учебников и монографий, в которых авторы явно указывают, какие предварительные знания необходимы для освоения материала, и в каком порядке рекомендуется читать книгу.

Еще одну трудность для людей, занимающихся самообразованием, представляет согласование систем обозначений. Изложение различных тем в рамках традиционного курса предполагает единство стиля, обозначений, последовательное именование и определение понятий. В то же время при самосто-

ятельном изучении материала обучающийся имеет дело с совокупностью документов (книг, статей, учебных пособий), в каждом из которых используется своя система обозначений и свои формулировки, не обязательно соответствующие формулировкам в других документах.

Обе эти проблемы, проблема учета взаимосвязей между разделами курса и проблема унификации обозначений и формулировок, встают и перед авторами, разрабатывающими учебные курсы, в особенности при подготовке курсов для дистанционного обучения, когда автор не может заранее с достаточной точностью оценить уровень подготовки слушателей. Стремление сделать курс самодостаточным может привести к включению в него необходимых фрагментов других курсов, изучение которых важно для понимания материала курса, но является излишним, если слушатель уже обладает соответствующими знаниями или изучил эти курсы ранее. Один и тот же материал может излагаться в нескольких курсах, ориентированных на разную целевую аудиторию или преследующих различные цели. При этом важно уменьшить трудозатраты преподавателей, которые составляют курсы, обеспечив возможность повторно использовать уже созданные учебные материалы.

Можно выделить два основных класса решений для управления материалами учебных курсов: объектно-ориентированные динамические учебные среды, такие как Moodle [7] и Sakai [15], и системы на основе технологии Wiki. В Wiki минимальной единицей содержания курса является страница. Логически связанные между собой страницы соединяются гиперссылками. Такая организация материала позволяет читателю быстро находить страницы, посвященные необходимым понятиям или утверждениям, однако последовательность изучения материала полностью определяется не автором курса, а самим читателем, который может пропустить важную информацию или, наоборот, потратить значительное время на изучение разделов, имеющих малое отношение к интересующей его теме. Стандартным, но только частичным решением является формирование оглавлений и гиперссылок, определяющих рекомендуемый порядок чтения страниц. Другим ограничением курсов, построенных с использованием технологии Wiki, является отсутствие формализации понятий, что затрудняет поиск материала (например, достаточно трудно выбрать из определенного раздела только определения или формулировки тео-

рем). Несмотря на эти ограничения, есть ряд известных учебников, созданных с использованием Wiki [14].

Объектно-ориентированные учебные среды (в частности, Moodle) позволяют авторам определять порядок следования элементов курса, но сами материалы курса остаются неструктурированными. Стандартное для этих систем крупноблочное деление материала (лекция, глава, задание) не позволяет явно ссылаться на отдельные элементы опубликованных материалов, что затрудняет решение таких задач, как формирование индивидуального плана для слушателя или формирование базового плана с дополнениями для разных курсов.

В обоих классах систем отсутствует возможность строить типизированные связи между отдельными понятиями курса и определять отношения порядка между элементами содержания курса, а не только между единицами хранения материалов. Эти возможности, в то же время, представляются важными для автоматизации работы преподавателей, составляющих курсы, и для повышения эффективности обучения. Использование структурированных на уровне содержания материалов позволит проверять логическую целостность курса, использовать в различных разделах курса согласованные системы обозначений и определений, упростить подготовку индивидуальных программ обучения и вариантов курса, обеспечить преемственность версий курса, читаемых в разные годы.

Представляемая работа посвящена построению системы хранения учебных материалов, учитывающую семантические особенности предметных областей. Это позволит расширить возможности поиска необходимых сведений и представить пользователю целостную систему учебных курсов, отражающую их взаимосвязи. Этот подход практически не реализуется в современных системах управления курсами, в связи с чем тема работы является актуальной. Для описания структуры учебных курсов используется онтологическая модель.

В результате выполнения работы была предложена методика построения учебных курсов на основе онтологической модели предметной области, и разработана информационная система для управления такими курсами.

1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Структура учебного курса

Конструирование содержания учебного материала представляет собой такое его упорядочение, при котором последовательность элементов содержания является основой, определяющей отношения между ними, т.е. построение содержания носит принцип систематичности.

Можно выделить набор логических отношений между элементами содержания: отношения следования, причинности, согласования, соподчинения, подобия и т. п. Например, в курсах по физике в подавляющем большинстве случаев мы будем иметь дело с причинно-следственными связями и объединением фактов по принципу подобия.

Среди логических построений выделяют систематичность линейную и систематичность структурную. Линейная логическая систематичность применяется в структурировании вузовских учебников академического типа и школьных учебниках для старших классов и представляет из себя такое построение материала, в котором использованы по возможности одинаковые логические отношения между элементами содержания, принадлежащими к одному предмету.

Структурное построение также использует по возможности одинаковые логические отношения прежде всего между элементами содержания, которые выбираются как основные из всей совокупности знаний по определенному предмету или одному из его разделов. Эти основные знания имеют линейное построение, а их качество и количество зависит от уровня обучения. По такому принципу часто строятся учебники и научные исследования.

В ряде работ по дидактике показано, что содержание учебного материала (образования) неоднородно по своей структуре. В связи с этим возможно различное деление содержания образования на элементы. Общепринятым считается разбиение содержания образования на знания, опыт деятельности в стандартных (обычных) ситуациях, опыт деятельности в нестандартных ситуациях (опыт творческой деятельности) и эмоционально-чувственный опыт, которые, в свою очередь, разбиваются на более мелкие элементы. Например, И.Я. Лернер [9] выделяет следующие виды знаний: основные термины и по-

нятия, факты повседневной деятельности и научные факты, основные законы науки, теории, методологические знания, оценочные знания, а А.В. Усова [9] выделяет такие элементы системы знаний: научные факты, понятия, законы, теории, научная картина мира. В целом различные авторы предлагают различные варианты систематизации и структурирования учебного материала. Логично предположить, что за основание систематизации и обобщения наиболее целесообразно принять основные структурные элементы системы научного знания, т. е. факты, понятия, явления, свойства, процессы, величины и т. д. При этом материал должен структурироваться таким образом, чтобы у учеников сформировалось не только предметное знание, но и методологическое знание, т. е. знание о структуре научного знания, способах его систематизации. Наиболее оптимальной структурой (систематизацией) знания (учебного материала) является интегральная дидактическая модель структуры учебного материала, в которой изложение материала происходит по циклу познания с позиций четырех аспектов описания явления: качественного, количественного, сущностного и прикладного.

При этом мы получаем возможность разлагать знание на отдельные структурные компоненты, без большого труда находить и вычленять в нем необходимые детали: отдельные факты, понятия, законы. Вместе с тем это вычленение не носит фрагментарный характер, что положительно сказывается на изучении материала.

Обучение с помощью компьютеров все больше и больше используется в образовании. Уже сейчас создаются виртуальные кафедры и виртуальные институты, обучаться в которых могут люди со всего мира. Уже созданы и продолжают создаваться обучающие системы по самым различным учебным курсам, причем не только по точным, но и по гуманитарным дисциплинам.

Для того, чтобы процесс обучения мог проводиться с помощью подобных обучающих систем, преподаватель вначале должен подготовить учебный материал, структурировать его и заложить в систему.

Отличительной особенностью решаемой задачи является сложная структура учебных материалов. С каждым курсом связан набор данных (свойств курса), описывающих сам курс, его положение в блоке дисциплин и учебном плане: название курса, его автор, требуемые для освоения курса знания, изучаемые в рамках курса разделы образовательных стандартов. Кроме это-

го, учебный курс характеризуется набором и последовательностью изучения тем, понятий, примеров, методов, задач, теорем, доказательств. Вводимые в рамках курса понятия определенным образом связаны между собой и с понятиями других курсов. Если для описания свойств курса можно применить хорошо известные методы объектной или реляционной декомпозиции, то использование этих же методов для описания структуры курса и связей между понятиями привело бы к необходимости постоянного изменения схемы базы данных. Такое изменение требует больших усилий и рассуждений в терминах структур данных, в то время как преподаватель составляет курс в терминах предметной области и методических терминах. С учетом этих особенностей принято решение строить реализацию системы хранения учебных материалов на основе онтологий.

1.2 Обзор технологий

1.2.1 Semantic Web

Основная идея Semantic Web заключается в том, чтобы сделать информацию, передаваемую в Web, более формализованной и удобной для машинного анализа, в частности, для идентификации и классификации. По мнению авторов технологии Semantic Web, это может достигаться посредством введения метаданных, которые должны сопровождать любую информацию и рассказывать о ее происхождении, формате и многом другом, что должно радикальным способом облегчить поиск информации в Web и ее обработку [6]. Основываясь на открытых стандартах, технологии Semantic Web позволяют описывать семантику произвольных данных, таких как содержание документов или код приложения. Основными в Semantic Web являются следующие технологии:

- глобальная схема имен (URI);
- модель описания данных (RDF);
- язык описания словарей (RDFS);
- средства описания связей между объектами данных (онтологии и язык их описания OWL).

Ключевым элементом технологий Semantic Web является система идентификации объектов. URI (Uniform Resource Identifier) — это идентификатор

какого-либо объекта (ресурса) в глобальной сети. Любой элемент, схема или модель данных семантической сети должны иметь собственный уникальный адрес. Сейчас используются два типа идентификаторов. Универсальный указатель ресурсов (Uniform Resource Locator, сокр. URL) — это URI, который, помимо идентификации ресурса, указывает на способ обращения с ресурсом путем описания способа доступа к нему или его положения в сети. Универсальное имя ресурса (Uniform Resource Name, сокр. URN) — это URI, который идентифицирует ресурс с помощью имени в определенном пространстве имен. Это позволяет ссылаться на ресурсе без использования информации об его расположении. Второй базовый компонент Semantic Web — это модель данных Resource Description Framework (RDF), которая позволяет объединить информацию из произвольных источников. Формат RDF наиболее полезен для обеспечения совместного использования информации, смысл которой может одинаково интерпретироваться различными программными агентами. RDF описывает предметную область в терминах ресурсов, свойств ресурсов и значений свойств. RDF-данные можно рассматривать как совокупность утверждений, включающих субъект, предикат и объект утверждения. Данные представляются в виде ориентированного графа, образуемого такими утверждениями. Следующий уровень в иерархии технологий Semantic Web занимает RDF Schema, язык описания словарей RDF-терминов. RDFS служит фундаментом для более богатых языков описания онтологий предметной области, которые позволяют адаптировать к Web логические системы и обеспечить семантическую обработку данных. Схема RDF представляет собой систему типов для Semantic Web и позволяет определить классы ресурсов и свойства как элементы словаря, в частности, описать, какие свойства относятся к каким классам.

1.2.2 Онтология и язык ее описания

Согласно принципам Semantic Web, процесс создания электронных документов разбивается на две части: создание собственно документа, содержащего некоторые термины, и создание его онтологии. Онтология может описываться различными средствами, и сегодня существует несколько языков описания онтологии, однако ввиду того, что в любой онтологии определяются термины и задаются логические связи между ними, точная семантика описываемых терминов и связей в различных языках будет одна и та же. Наиболее

удачным языком является Ontology Web Language (OWL), который описывает онтологию в формате XML.

OWL добавляет к RDF и RDFS дополнительные возможности для описания свойств и классов: отношения между классами, равенство, новые типы и характеристики свойств, перечисляемые классы. OWL предоставляет три подмножества, имеющие различную степень детализации.

Формальная семантика OWL определяет логический вывод на основе онтологий, т. е. способ получения фактов, которые не представлены в онтологии непосредственно. Вывод могут базироваться на анализе одного документа или множества документов, распределенных в Web. Последнее достигается за счет возможности связывания онтологий, включая прямой импорт информации из других онтологий.

Чтобы создать онтологию, которая может однозначно интерпретироваться и использоваться программными агентами, задействуется синтаксис и формальная семантика OWL. На практике создание онтологий начинается с формирования иерархии классов понятий, составляющих предметную область. Для того, что бы понятия предметной области были наполнены определенным смысловым содержанием, они должны характеризоваться конкретными наборами свойств и состоять в определенных связях друг с другом. Эту задачу в языке OWL решают механизмы свойств и ассоциированных с ними ограничений. Свойства подразделяются на два вида: свойства-характеристики и свойства-связи. Свойства-характеристики описывают классы, свойства-связи ассоциируют классы друг с другом. На свойства накладываются ограничения двух типов: глобальные и локальные. К глобальным ограничениям относятся домены, то есть классы, экземпляры которых могут обладать этими свойствами, и диапазоны (классы, экземпляры которых могут выступать в качестве значений этих свойств). Локальные ограничения накладываются на свойства в рамках определенного класса и могут еще более сужать диапазоны для свойств в рамках этого класса, определять мощность свойств и их виды.

1.2.3 Topic Maps

Технологии Topic Maps и RDF возникли достаточно давно и до некоторого времени развивались независимо друг от друга. Они поддерживались, с одной стороны, ассоциацией ISO, а с другой — консорциумом W3C. Несмотря

на то, что эти семантические технологии очень похожи, они изначально были предназначены для различных целей. Технологии Topic Maps создавались с целью обеспечения наиболее качественного поиска по содержимому Web, а методология RDF предоставляет, прежде всего, возможность структурного описания ресурсов с помощью метаданных и логических связей. Синтаксис Topic Maps описывается с помощью языка XTM (XML Topic Maps), который позволяет адаптировать стандарт topic map (ISO 13250) для совместного использования с XML и другими стандартами Web. Язык XTM предоставляет синтаксис описания Topic Maps, основанный на XML [30]. Аналогом RDFS в технологии Topic Maps является Ontopia Schema Language (OSL). В отличие от RDFS, для языка описания онтологий OWL не существует аналога в технологии Topic Maps, так как OWL расширяет возможности RDFS и не выражается через него.

1.2.4 Learning Object Metadata

Целью стандарта LOM [19] (Learning Object Metadata — метаданные учебного объекта) является упрощение поиска, рассмотрения и использования учебных объектов учителями, инструкторами, предоставить возможность автоматического доступа к учебным объектом, а также облегчение совместного использования таких объектов путем создания каталогов и хранилищ. Стандарт предлагает базовую схему, которая может использоваться для создания практических разработок, например, с целью автоматического адаптивного назначения учебных объектов тем или иным агентам программного обеспечения. Стандарт не определяет, каким способом обучающие системы будут представлять или использовать метаданные учебных объектов. Учебные объекты описываются элементами данных, сгруппированными в категории. Базовая схема LOM версии 1.0 определяет девять таких категорий [8].

1. Общая категория объединяет информацию об учебном объекте в целом.
2. Категория жизненного цикла группирует элементы об истории и текущем состоянии учебного объекта и тех, кто влиял на него в ходе эволюции.
3. Категория мета-метаданных содержит информацию о метаданных.

4. Техническая категория группирует технические требования и характеристики учебного объекта.
5. Образовательная категория объединяет образовательные и педагогические характеристики.
6. Категория прав содержит данные об интеллектуальной собственности и условиях использования.
7. Категория связей (отношений) определяет понятия, определяющие взаимосвязи между данным и иными учебными объектами.
8. Категория аннотаций представляет комментарии к учебному использованию объекта и данные о создателях этих комментариев.
9. Классификационная категория определяет место данного объекта в пространстве той или иной классификационной схемы.

Вместе эти категории образуют базовую схему LOM. При использовании классификационной категории возможны различные типы расширений этой схемы.

Категории группируют элементы данных. Модель данных LOM имеет иерархическую структуру и включает как агрегаты элементов данных, так и простые элементы данных (листья на иерархическом дереве). В базовой схеме версии 1.0 только простые элементы имеют индивидуальные значения, определенные путем ассоциации с пространством значений и типом данных. Агрегаты индивидуальных значений не имеют.

Для каждого элемента данных базовая схема определяет:

- имя;
- объяснение (explanation) — определение элемента данных;
- размер (size) — число разрешенных значений;
- порядок (order) — если порядок значений является важным;
- пример.

Для простого элемента также определены:

- пространство значения (value space) — набор разрешенных значений, обычно в форме словаря или ссылки на другой стандарт;
- тип данных (datatype) — LangString, DateTime, Duration, Vocabulary, CharacterString или Undefined.

Ни один элемент данных не является обязательным, что означает, что формату LOM соответствуют любые сочетания элементов данных.

1.3 Обзор существующих решений

Существует несколько типов решений для хранения учебных материалов, которые можно использовать для создания учебных курсов. Основными из них можно считать два класса: системы управления обучением и системы на основе технологии Wiki. Wiki позволяет пользователям редактировать любую страницу или создавать новые страницы, поддерживает связи между разными страницами путем создания гиперссылок, в том числе ссылок на еще не существующие страницы. Системы управления обучением также позволяют создавать и добавлять материалы курса, но в них обычно не поддерживаются связи между разделами курса, а также отсутствует возможность создавать модифицированные версии курсов, например, соответствующие индивидуальным планам слушателей.

Помимо традиционных текстовых Wiki выделяется класс семантических Wiki. Существуют два типа семантических Wiki-сред, тексто-ориентированные и логико-ориентированные. Тексто-ориентированные семантические Wiki расширяют классические Wiki-среды семантическими аннотациями, формируя набор отношений между текстовыми элементами. Целью таких семантических Wiki является не работа с онтологиями, а предоставление формального каркаса для Wiki-статей. Известными примерами таких систем являются Semantic MediaWiki, KiWi (дальнейшее развитие IkeWiki) и KnowWE. Логически-ориентированные семантические Wiki являются платформами для разработки онтологий. Их цель состоит в том, чтобы упростить и ускорить накопление, хранение и анализ формальных знаний. Примерами таких сред являются AceWiki и OntoWiki [16].

В таблице 1.1. представлены результаты сравнительного анализа функциональных возможностей различных семантических Wiki [28].

Таблица 1.1.

Сравнение семантических Wiki

Критерий сравнения	IkeWiki	Makna	OntoWiki	OpenRecord	SMW	SemperWiki	WikSAR
Редактор онтологий	+	—	—	+	—	—	—
WYSIWYG редактор	+	—	+	+	+	—	—
Контексто-зависимая навигация	+	+	+	—	+	+	+
Различное отображение	—	—	+	—	—	—	+
Фасетный поиск	—	—	+	—	—	—	—
Визуализация графических связей	—	—	—	—	—	—	+
Браузер онтологий	+	—	+	—	—	—	—
Встроенный язык запросов	—	—	—	—	+	—	+
Полнотекстовый поиск	+	+	+	+	+	+	+
Шаблоны запросов	—	+	—	—	—	—	—
Импорт онтологий	—	+	—	—	+	—	—
Экспорт онтологий	—	+	—	—	+	—	—
Отслеживание изменений	+	—	+	+	+	—	—
Распространенность	—	—	+	—	—	—	—
Возможность оставлять комментарии	—	—	+	—	—	—	—

В результате сравнения возможностей наиболее популярных реализаций семантических Wiki было выяснено, что наиболее подходящими для использования в рассматриваемой области являются Semantic MediaWiki и OntoWiki. Их преимущества перед остальными заключаются в наличии удобного WYSIWYG-редактора, контекстно-зависимой навигации, визуализации графических связей или браузера онтологий и возможности отслеживания изменений.

1.3.1 Semantic MediaWiki

Semantic MediaWiki (SMW) — это расширение Wiki-движка MediaWiki, которое помогает искать, организовывать, снабжать теги, просматривать, обрабатывать и разделять содержимое Wiki. Semantic MediaWiki была анонсирована в 2005 году, и в настоящее время имеет более десяти разработчиков и используется на сотнях сайтов. Также на сегодняшний день реализовано большое количество семантических расширений, призванных расширить возможности редактирования и представления данных.

Одним из ограничений традиционных Wiki является слабое развитие технологий автоматической обработки данных, которые особенно проявляются при большом количестве представленных в системе страниц. Перечислим некоторые проблемы, возникающие из-за отсутствия эффективных методов обработки данных:

- Несогласованность данных на разных страницах.
- Трудоемкость повторного использования данных.
- Трудность или невозможность отбора страниц, объединенных общими свойствами.
- Отсутствие развитого поиска.

Semantic MediaWiki позволяет хранить в Wiki структурированные, пригодные для машинной обработки данные, использование которых позволяет решать эти проблемы. Для хранения структурированных данных SMW поддерживает дополнительные элементы разметки текста, так называемые семантические аннотации. Они позволяют упростить структуру Wiki-сайта, предоставляя следующие возможности [26]:

– Автоматически генерируемые списки. Автоматическая генерация позволяет поддерживать актуальность списков, отбирать данные, удовлетворяющие набору условий, динамически изменять параметры оформления.
– Визуальное отображение информации. Расширения Wiki, такие как Semantic Result Formats или Semantic Maps, дают возможность отображать информацию в виде календарей, диаграмм, графиков, карт, обеспечивая намного более наглядное и интуитивное представление, чем простые списки.
– Типизированные атрибуты. Каждая страница может быть снабжена набором аннотаций, тип которых указывается явным образом. Это позволяет решить задачу каталогизации страниц по нескольким признакам. В традиционных реализациях Wiki для поддержки нескольких вариантов классификации используется система категорий, которая при увеличении количества используемых для классификации признаков становится чрезмерно сложной.
– Удобство ввода структурированной информации. Администраторы системы могут создавать формы ввода и редактирования структурированных данных, что позволяет контролировать правильность ввода данных и снижает трудозатраты пользователей.
– Поиск информации. Пользователи могут искать определенную информацию, создавая свои собственные запросы.
– Согласованность данных на разных языках. Использование структурированных данных позволяет автоматически обнаруживать некоторые случаи рассогласования между версиями одних и тех же данных на разных языках.
– Расширенные возможности экспорта данных и поддержи внешних интерфейсов. Структурированные данные могут экспортироваться из SMW в одном из стандартных форматов (CSV, JSON, RDF). Это позволяет внешним приложениям, знающим о структуре данных в системе, стандартным образом выполнять запросы к системе. На основе механизма запросов и экспорта данных можно объединить несколько экземпляров SMW, избегая необходимости ручной синхронизации данных.

– Поддержка импорта данных. Системы, построенные на базе SMW, могут собирать данные, экспортируемые другими системами, используя технологии web-сервисов и RDF.

1.3.2 Onto Wiki

Представителем второго класса систем является Onto Wiki [22], разрабатываемая в университете Лейпцига. В отличие от SMW, в которой формализованные данные играют роль дополнительных атрибутов, добавляемых к основному слабоструктурированному содержимому, Onto Wiki изначально спроектирована для хранения только формализованных данных, представленных в формате RDF. Для отображения данных в web-браузере формируется удобное для восприятия человеком текстовое представление, а также структурированные формы для ввода и редактирования данных. Внешние программы взаимодействуют с Onto Wiki, используя сериализованное представление RDF и язык запросов SPARQL.

Предоставляемые системой средства навигации по данным включают средства отображения таксономий и иерархий, инструменты фасетной навигации, поиск по атрибутам и полнотекстовый поиск. Эти средства можно сочетать между собой, что позволяет уточнять условия поиска. Запросы, выполняемые с помощью каждого из этих инструментов, преобразуются системой в запрос на языке SPARQL. Результаты запросов отображаются в виде списка, а также могут быть преобразованы в дополнительные представления, примерами которых являются карты или календари. Набор поддерживаемых форм представления данных может быть расширен администраторами системы.

Каждой форме отображения данных в Onto Wiki соответствует автоматически генерируемая форма ввода и редактирования данных. Таким образом, пользователи могут пополнять хранилище данных Onto Wiki без необходимости изучать RDF или другие используемые в системе технологии. Новые данные могут также импортироваться в систему со страниц web-сайтов. Предусмотрены также средства экспорта данных, позволяющие отображать на сторонних сайтах актуальные версии хранимых данных. Различные системы, построенные на основе OntoWiki, могут обмениваться данными как между собой, так и с другими информационными системами, поддерживающими стандарты RDF и FOAF.

Onto Wiki позволяет не только пополнять набор хранимых данных, но и менять структуру этих данных, причем система упрощает внесение необходимых изменений, предоставляя пользователям расширяемый набор предопределенных алгоритмов модификации онтологий. Это позволяет отражать в системе те изменения во взгляде на предметную область, которые естественным образом происходят по мере приобретения пользователем опыта и знаний.

1.3.3 Moodle

Moodle (англ. Modular Object-Oriented Dynamic Learning Environment — модульная объектно-ориентированная динамическая учебная среда) — свободная система управления обучением (LMS), распространяющаяся под свободной лицензией GNU GPL. Основными вариантами применения системы является организация взаимодействия работы студентов и преподавателей, совместное решение учебных задач, публикация учебных материалов для дистанционного и очного обучения. Также Moodle позволяет хранить все материалы, которые создают студенты в процессе обучения.

При подготовке и проведении занятий в системе Moodle преподаватель использует набор стандартных элементов курса, в который входят глоссарий, ресурс, задание, форум, документ Wiki, урок, тест. Варьируя сочетания различных элементов курса, преподаватель организует изучение материала таким образом, чтобы формы обучения соответствовали целям и задачам конкретных занятий. Опишем подробнее эти стандартные элементы.

Глоссарий представляет собой совокупность словарных статей, соответствующих употребляемым в курсе терминам. Глоссарий может пополняться не только преподавателем, но и студентами. Занесенные в глоссарий термины отображаются в материалах курса в виде гиперссылок. Поддерживается использование глоссариев, соответствующих каждому отдельному курсу, и общего для всех курсов глобального глоссария.

Ресурсами называются неструктурированные учебные материалы: тексты, иллюстрации, web-страницы, аудио или видео файлы. Для редактирования некоторых типов ресурсов предусмотрен визуальный редактор.

Выполнение задания — это вид деятельности студента, результатом которой обычно становится создание и загрузка на сервер файла или создание

текста непосредственно в системе Moodle при помощи встроенного визуального редактора. Преподаватель может оперативно проверить опубликованные результаты, прокомментировать их и, при необходимости, предложить доработать. Результаты выполнения заданий могут быть опубликованы для совместного обсуждения. Результаты выполнения заданий можно сдавать неоднократно, что позволяет итеративно улучшать качество решения и добиваться полного решения задачи.

Построение курса в виде набора уроков позволяет организовать пошаговое изучение учебного материала. Совокупность учебных материалов разбивается на дидактические единицы, каждая из которых сопровождается набором вопросов, позволяющих проверить усвоение материала. Результаты изучения материала могут автоматически оцениваться с помощью тестов. Тесты поддерживают разные типы вопросов: вопросы в закрытой или открытой форме, вопросы с числовыми ответами или выбором соответствия. Подготовленные преподавателем тестовые задания сохраняются в базе данных и могут быть использованы в различных курсах. Варианты тестов могут формироваться как путем явного выбора задач преподавателем, так и с помощью случайного выбора заданий из определенного раздела.

1.3.4 Sakai

Sakai — система сетевого и дистанционного обучения, виртуальная среда для организации обучения и совместной работы. Sakai представляет собой набор программных инструментов, предназначенных для того, чтобы помочь преподавателям и студентам в поддержке очного учебного процесса или организации дистанционного обучения; кроме того, Sakai может служить средой для взаимодействия исследовательских групп. Для организации совместной работы в Sakai есть набор инструментов, обеспечивающих коммуникацию и групповую деятельность как на рабочем месте, так и удаленно. Используя браузер, пользователи могут выбирать набор инструментов на сайте курса, изменяя таким образом функциональность для своих целей. Для того чтобы использовать систему, не требуется никаких специальных знаний, таких, как знание HTML. Приведем несколько примеров сайтов, которые можно создать с помощью Sakai.

- Персональный сайт пользователя создается автоматически при регистрации пользователя, является местом хранения личных ресурсов (аудио-, видеофайлы, изображения, текстовые документы) и рабочим пространством (персональные настройки системы, резюме, персональный wiki-журнал), а также консолидирует всю необходимую информацию (в частности, автоматически ретранслируются объявления со всех сайтов пользователя и формируется индивидуальное расписание).
- Сайт учебного курса используется для управления материалами каждого курса. На сайте курса студенты могут ознакомиться с программой и расписанием курса, получают доступ к материалам занятия, проходят тесты и письменные экзамены, выполняют другие задания, оправляя свои работы преподавателю. Sakai поддерживает проведение интерактивных онлайн-занятий с использованием аудио- и видео-конференц связи и виртуальной классной доски.
- Сайт-проект предназначен для организации взаимодействия группы пользователей, например, кафедры, лаборатории, участников исследовательского проекта. Сайт предоставляет участникам возможность размещать объявления и осуществлять рассылку, предоставлять доступ к необходимым электронным ресурсам, таким как документы или ссылки на другие веб-сайты.
- Сайт-портфолио используется для публикации результатов работы. С помощью этой разновидности сайта можно контролировать качество выполнения как отдельных компонентов учебного процесса, так и учебного плана в целом.

Все вышеперечисленные типы сайтов могут служить точками доступа ко всем типам библиотечных/электронных ресурсов: лицензионных электронных баз данных (научных журналов, электронных книг), электронных каталогов традиционной библиотеки вуза, локальных электронных коллекций подразделения.

Система Sakai написана на языке Java и распространяется под свободной лицензией Education License. Для работы системы необходима Java 1.5 SE; в качестве веб-контейнера используется Apache Tomcat 5.5; для хранения данных используется реляционная база данных (на данный момент поддерживается MySQL 4.1.12 и выше с InnoDB, Oracle9i и выше). Система реализует интерфейсы, описанные в группе открытых стандартов OKI OSID (Open

Knowledge Initiative Open Service Interface Definitions). Sakai представляет собой расширяемую систему, позволяющую разработчикам создавать собственные инструменты и сервисы, которые могут быть развернуты в любой совместимой в Sakai оболочке. Для обеспечения единообразия внешнего вида при создании элементов представления используются библиотеки тэгов. Система легко масштабируема, поддерживает распределенную работу.

Сравнительный анализ систем Moodle и Sakai показал, что обе системы обладают сравнимыми возможностями, но Moodle имеет ряд преимуществ, делающих его внедрение и применение более простым:

- Moodle включает более подробную документацию для каждой из ролей: администраторов, преподавателей, обучающихся и разработчиков.
- Сообщество активных пользователей и разработчиков Moodle включает большее число участников, предоставляющих информацию о Moodle и обменивающихся знаниями и мнениями о решении различных вопросов в данной системе.
- Пользовательский интерфейс Moodle более интуитивно понятен и прост в освоении даже для начинающих пользователей, одновременно являясь более гибким для опытных.
- Использование Moodle требует меньшего количества вычислительных ресурсов.

В завершении обзорной части следует отметить что во всех рассмотренных выше системах (как базирующихся на технологиях Semantic Wiki, так и широко распространенных LMS) отсутствует возможность задавать типизированные связи между понятиями курса, а также отношения порядка для изложенного в курсе материала. Отсутствие таких связей приводит к необходимости неформально указывать рекомендуемый порядок изучения материала и требования, предъявляемые к слушателям курса, что затрудняет повторное использование материала в других курсах. Другим препятствием для повторного использования материала является способ деления материала на отдельные фрагменты. В системах на основе Wiki возможно создание отдельных статей, включающих относительно небольшие фрагменты материала (определения, теоремы, задачи). Тем не менее, такое мелкое деление затрудняет объединение элементов для последовательного изложения, поскольку Wiki не предусматривает средств навигации с учетом порядка следования материала в курсе.

Системы управления учебными материалами обычно не позволяют делить материал на отдельные повторно используемые элементы (исключениями могут быть определения и задачи), из-за чего авторы курса обычно ограничиваются делением материала на отдельные лекции. Все это делает весьма сложным подготовку индивидуальных учебных планов, а также использование опубликованного материала для самостоятельного обучения. При использовании Wiki самостоятельно обучающийся студент сталкивается с необходимостью самостоятельно определять взаимосвязи и целесообразный порядок изучения материала, представленного в виде неупорядоченной совокупности статей. В то же время ограниченная поддержка поиска и слабая структурированность материала в современных LMS приводят к тому, что студент оказывается вынужден последовательно изучать материалы курса, чтобы найти в них отдельные необходимые ему сведения.

Таким образом, актуальной является задача построения системы управления материалами курса на основе семантических связей. Разрабатываемая система должна позволять определение зависимостей между материалами на основе уровня подготовки слушателя, различных требованиям к излагаемому материалу, а также учитывать особенности групп слушателей и факультетов, упростить создание курсов одной предметной области или нескольких курсов, основывающихся на общем наборе учебно-методических материалов.

2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

2.1 Постановка задачи

Предметом данной работы является проектирование и реализация системы управления учебными материалами на основе семантических моделей предметных областей. В ходе выполнения работы необходимо:

- Разработать базовую структуру учебных материалов, определяющую множество типов учебных материалов и возможные отношения между ними.
- Построить базовую семантическую модель курса, включающую множество используемых материалов и порядок изложения материала.
- Реализовать прототип системы управления учебными материалами, позволяющий строить курсы на основе их семантической модели, включать в курс учебные материалы, проверять зависимости между материалами и формировать текстовое представление курса для слушателей.

2.2 Формальное описание учебных курсов

2.2.1 Структура учебных материалов

Для описания структуры учебных материалов необходимо определить, какие типы учебных материалов используются для изложения предмета в данной предметной области, и задать возможные отношения между материалами. Используемый в данной работе метод основан на следующей формальной модели.

Разработчик структуры учебных материалов задает два множества: T — множество типов учебных материалов, и R — множество отношений между материалами. Например, при описании структуры материалов для школьной геометрии множество T включает элементы «определение», «аксиома», «лемма», «теорема», «следствие», «доказательство», «задача», «решение», «рисунок», а в множество R входят элементы «использует», «доказывает», «является решением», «иллюстрирует», «применяется для решения».

Структура учебных материалов описывается мультиграфом $S = \langle T, E \rangle$, где $E \subseteq T \times R \times T$ — множество ребер графа, которое определяет, материалы каких типов могут быть связаны каждым из отношений.

Соответствующий этой структуре набор учебных материалов описывается мультиграфом $D = \langle M, Q \rangle$, где M — множество учебных материалов, а $Q \in M \times R \times M$ — множество отношений между материалами. При этом выполняются следующие свойства.

1. Каждый учебный материал имеет единственный тип из множества типов T: $M = \bigcup_{t \in T} M_t \wedge \forall i \in T . \forall j \in T . [M_i \cup M_j = \emptyset]$.
2. В мультиграфе, описывающем набор учебных материалов, ребро, связывающее два материала, может быть проведено только тогда, когда оно допустимо для соответствующих этим материалам типов: $\forall a \in M_x . \forall b \in M_y . [(a, r, b) \in Q \Rightarrow (x, r, y) \in E]$.

Элементы множества M представляют собой тройки $(dsc, cnt, type)$, где dsc — краткое описание, характеризующее учебный материал, cnt — содержание учебного материала, которое может быть представлено в виде текста или ресурса любого другого вида, и $type \in T$ — тип материала.

В любой структуре материалов множество T обязательно должно содержать элемент $t_0 \in T$, соответствующий отношению «использует». На основе этого отношения строится частичный порядок изучения материала.

2.2.2 Структура учебных курсов

С точки зрения формальной модели учебный курс представляет собой совокупность учебных материалов, между которыми определено отношение частичного порядока, определяющее последовательность изложения материала, а также набор учебных материалов, предварительное знакомство слушателя с которыми предполагается авторами курса.

Курс описывается как

2.2.3 Другие подходы к описанию курсов

Один из подходов к представлению структуры учебного материала приведен в [5]. Опишем формально этот подход и рассмотрим, каким образом можно использовать его для описания зависимостей между материалами.

Пусть множество из n курсов: $\Pi = \{\Pi_i\}; i = 1,...,n$. Каждый курс множества Π делится на разделы. Пусть n_i — количество разделов в i-м курсе. Обозначим множество разделов курса i как $P_i = \{P_{ij}\}, i = 1,...,n; j = 1,...,n_i$, где i — номер курса, j — номер раздела внутри курса.

Каждый раздел множества P_i делится на темы. Пусть n_{ij} - количество тем в j-м разделе i-го курса. Тогда можно ввести множество тем: $T_{ij} = \{T_{ijk}\}, i = 1,...,n; j = 1,...,n_i; k = 1,...,n_{ij}$, где i — номер курса, j — номер раздела курса, k — номер темы внутри раздела.

Под темой будем понимать минимальную целостную часть учебного материала. Для каждой темы можно задать, какие знания и умения должен получить обучаемый, изучив данную тему. Знания определяются терминами, или определениями. В рамках каждой темы могут изучаться несколько определений. Пусть $n^{\mathrm{З}}_{ijk}$ — их количество для указанной темы. Тогда введем множество определений в теме:

$$\mathrm{З}_{ijk} = \{\mathrm{З}_{ijkl}\}, i = 1,...,n; j = 1,...,n_i; k = 1,...,n_{ij}; l = 1,...,n^{\mathrm{З}}_{ijk}$$

где i — номер предмета, j — номер раздела внутри предмета, k — номер темы внутри раздела, l — номер определения внутри темы.

Каждый термин имеет вес $n^{\text{З}}_{ijkl} \in 0{,}1$. Чем выше вес термина, тем большее значение он имеет при изучении материала, изложенного в теме.

Для каждой темы задается, какие умения должен получить обучаемый. Пусть $n^{\text{У}}_{ijk}$ — их количество для указанной темы. Тогда введем множество умений для темы:

$$\text{У}_{ijk} = \text{У}_{ijkl}, i = 1,...,n; j = 1,...,n_i; k = 1,...,n_{ij} l = 1,...,n^{\text{У}}_{ijk}$$

где i — номер предмета, j — номер раздела внутри предмета, k — номер темы внутри раздела, l — номер умения внутри темы.

Каждое умение имеет вес $n^{\text{У}}_{ijkl} \in \{0{,}1\}$. Чем выше вес умения, тем большее значение оно имеет при изучении материала темы.

Исходя из этого, можно определить: $B_{ijk} = K_{\text{З}} \sum\limits_{l=1}^{n^{\text{З}}_{ijk}} B^{\text{З}}_{ijkl} + K_{\text{У}} \sum\limits_{l=1}^{n^{\text{У}}_{ijk}} B^{\text{У}}_{ijkl}$

1. вес темы в разделе: $B_{ij} = \sum\limits_{k=1}^{n_{ij}} B_{ijk} = \sum\limits_{k=1}^{n_{ij}} (K_{\text{З}} \sum\limits_{l=1}^{n^{\text{З}}_{ijk}} B^{\text{З}}_{ijkl} + K_{\text{У}} \sum\limits_{l=1}^{n^{\text{У}}_{ijk}} B^{\text{У}}_{ijkl})$, где $\text{K}_{\text{з}}$ и $\text{K}_{\text{у}}$ — весовые коэффициенты.
2. вес раздела в курсе: $B_i = \sum\limits_{j=1}^{n_i} B_{ij} = \sum\limits_{j=1}^{n_i} \sum\limits_{k=1}^{n_{ij}} (K_{\text{З}} \sum\limits_{l=1}^{n^{\text{З}}_{ijk}} B^{\text{З}}_{ijkl} + K_{\text{У}} \sum\limits_{l=1}^{n^{\text{У}}_{ijk}} B^{\text{У}}_{ijkl})$
3. вес курса: $B = \sum\limits_{i=1}^{n} B_i$.

Можно также определить нормированные веса: $e^{\text{З}}_{ijkl} = \frac{B^{\text{З}}_{ijkl}}{\sum\limits_{l=1}^{n^{\text{З}}_{ijk}} B^{\text{З}}_{ijkl}}; \sum\limits_{l=1}^{n^{\text{З}}_{ijk}} e^{\text{З}}_{ijkl} = 1$

1. термина в теме: $e^{\text{У}}_{ijkl} = \frac{B^{\text{У}}_{ijkl}}{\sum\limits_{l=1}^{n^{\text{У}}_{ijk}} B^{\text{У}}_{ijkl}}; \sum\limits_{l=1}^{n^{\text{У}}_{ijk}} e^{\text{З}}_{ijkl} = 1$
2. умения в теме: $e_{ijk} = \frac{B_{ijk}}{B_i j}; \sum\limits_{k=1}^{n_{ij}} e_{ijk} = 1$
3. темы в разделе: $e_{ij} = \frac{B_{ij}}{B_i}; \sum\limits_{j=1}^{n_i} e_{ij} = 1$
4. раздела в курсе: $e_i = \frac{B_i}{B}; \sum e_i = 1$

5. курса: $e_i = \frac{B_i}{B}; \sum_{i=1}^{n} e_i = 1$

Таким образом, на основе полученных коэффициентов можно определить порядки важности:

- знаний в теме.
- умений в теме.
- тем в разделе.
- разделов в курсе.
- курсов.

Если суммарный вес нескольких подряд идущих тем раздела превосходит критический вес $\mathrm{B}_{\text{крит}}$, то после последней темы этой последовательности следует проводить контроль знаний обучаемого. Если вес какой-либо темы $\mathrm{B}_{ijk} > \mathrm{B}_{\text{крит}}$, то после этой темы также необходимо провести контрольные мероприятия. Величина критического веса $\mathrm{B}_{\text{крит}}$ определяется преподавателем-экспертом.

Введем обозначение $\tau = \tau_m, m = 1,...,n^\tau$ для общего множества всех тем всех курсов. Здесь $n^\tau = \sum_{i=1}^{n} \sum_{j=1}^{n_i} n_{ij}$ — общее количество тем во всех разделах всех курсов.

Множества Π, P и T связаны со множеством τ отношением R_τ, с помощью которого можно определить, какому курсу $\Pi_i \in \Pi$, разделу $\mathrm{P}_{ij} \in P_i$ и теме $T_{ijk} \in T_{ij}$ принадлежит элемент $\tau_m \in \tau$. Одно из возможных соотношений, связывающих эти четыре множества, может быть следующим. Пронумеруем все элементы всех четырех множеств. Пусть i — номер курса, j — номер раздела в курсе, k — номер темы в разделе, m — номер элемента в множестве τ. Тогда можно записать следующее соотношение: $m = 10000 \times i + 100 \times j + k$. Оно задает линейный порядок на множестве тем.

Недостатком данного соотношения является ограничение на количество курсов, разделов в курсе и тем в разделе (не более 100). Однако данная проблема может быть решена введением других коэффициентов.

Иерархию тем будем строить по множеству τ. Введем отношение: $\tau_j S_T \tau_i$ — для знания материала темы τ_j необходимо знание материала темы τ_i. Отношение S_T обладает следующими свойствами:

1. Рефлексивность: для знания τ_i необходимо знание τ_i.

2. Транзитивность: $\tau_i S_T \tau_j \& \tau_j S_T \tau_k \rightarrow \tau_i S_T \tau_k$.
3. Антисимметричность: $\tau_i S_T \tau_j \& \tau_j S_T \tau_k \rightarrow \tau_i = \tau_j$.

Таким образом, получается частичный порядок. На его основании можно построить граф порядка изучения тем, на рис. 2.1. в виде графа отображены обязательные зависимости, а также желательные зависимости (пунктирные стрелки), использованные в книге [2].

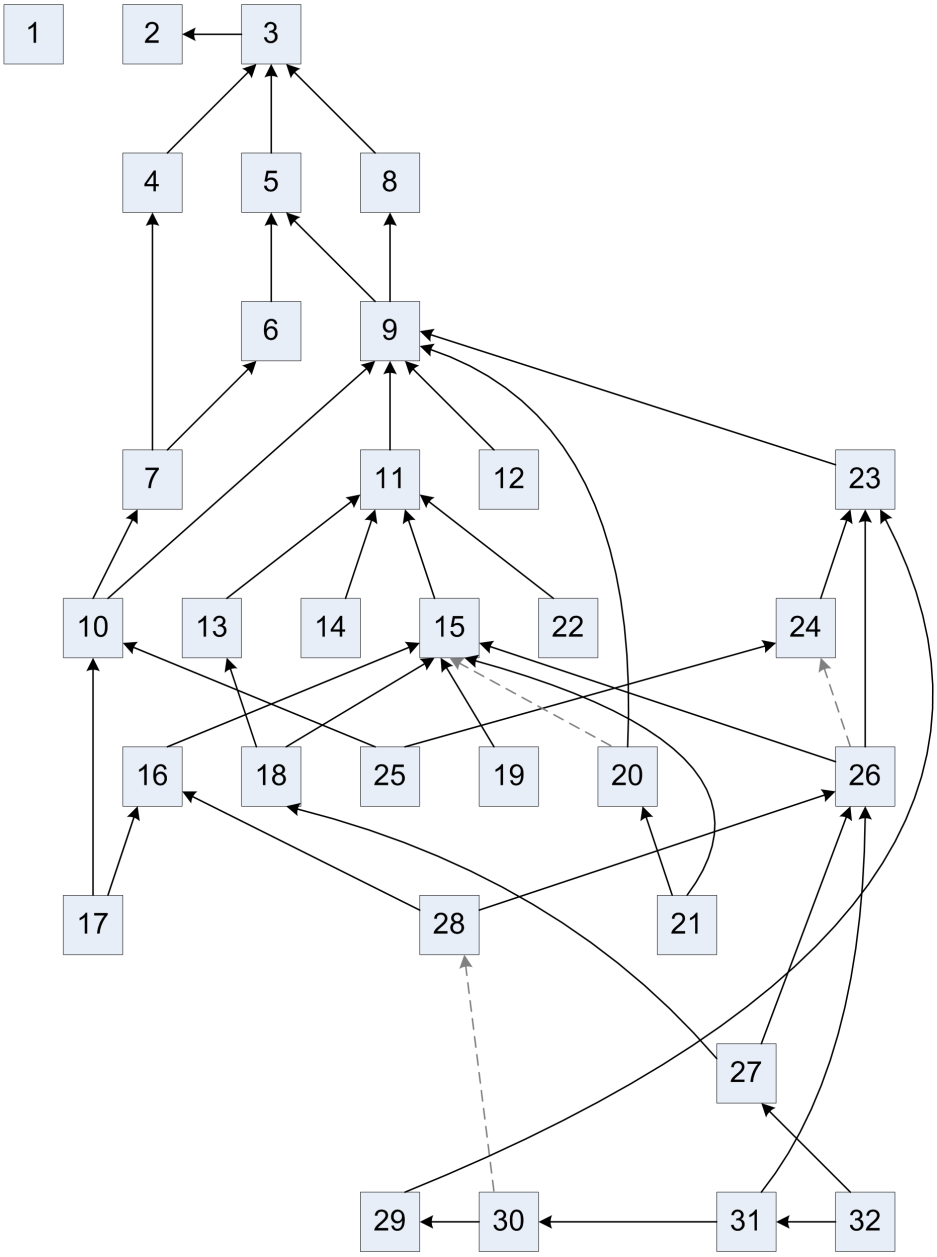


Рис. 2.1. **Зависимости между главами**

2.3 Создание дистанционного курса

В работе [12] описана технологическая схема создания курса дистанционного обучения, включающая пять основных этапов:

1. Подготовительный этап.
2. Разработка проекта.
3. Этап производства курса.
4. Этап тестирования и пробного проведения курса (апробации).
5. Этап усовершенствования и тиражирования курса.

На первом, подготовительном, этапе определяется целевая аудитория курса и решаемые в рамках курса задачи, оцениваются затраты на разработку и поддержку курса, выбираются методы публикации курса. Кроме этого, необходимо проанализировать другие доступные целевой аудитории учебные материалы в данной предметной области и оценить ожидаемый экономический эффект от внедрения курса.

На этапе разработки проекта производится подробная разработка структуры курса, определяются основные разделы и компоненты курса, выбираются технические средства и форматы, необходимые для реализации всех элементов курса. На этом же этапе организуется совместная работа участников. Авторы [12] отмечают, что для обкатки технологии полезно разработать небольшого фрагмента курса, который, при этом, содержал бы все используемые в курсе типы элементов (текст, рисунки, аудио- и видеоматериалы, тесты). На этом этапе формируется календарный график работ и технологическая документация.

Этап производства курса является основным и занимает наибольшее время. На этом этапе разрабатывается содержание курса, создаются учебные материалы (в том числе мультимедийные материалы и программы), происходит отладка курса для каждого используемого способа публикации. Готовятся руководства по изучению курса для студентов, преподавателей и технических специалистов.

На этапе апробации проводится комплексное тестирование курса, поиск ошибок, анализируется эффективность курса, формируется стратегия продвижения курса. Для этого производится набор тестовых групп слушателей.

Последний этап технологической цепочки завершается публикацией курса, его тиражированием. На этом этапе производится анализ результатов апробации курса, исправляются обнаруженные ошибки. При этом возможно изменение содержания курса, создание новых учебных материалов. По окончании работы по усовершенствованию курса производится окончательное тиражирование материалов (публикация на web-сервере, выпуск компакт-дисков, полиграфической продукции).

Основываясь на таблице временных затрат [12] на подготовку курса, можно сделать вывод, что наибольшее время занимает этап разработки курса. Предоставив авторам курса возможность повторно использовать подготовленный материал для других похожих курсов, можно существенно сократить время разработки нового курса.

2.4 Выбор средств разработки

В качестве основы для реализации данной системы решено использовать один из существующих фреймворков Semantic web, которые позволяют описывать модель хранения данных в приложении. Учитывая специфику данной задачи, были рассмотрены следующие среды:

- CubicWeb — фреймворк для создания веб-приложений на языке програмирования Python, активно развивающийся и следующий объектно-ориентированной парадигме [29].
- Ontopia — фреймворк для создания приложений, основанных на технологии Topic Maps [25].
- Web-фреймворк Django с плагином DjangoRDF [20].

2.4.1 DjangoRDF

DjangoRDF предоставляет средство для работы с RDF из приложений Django, при этом существующие приложения не требуют модификации разработанных моделей данных и дополнительной настройки хранилищ данных. В качестве языка запросов моделей данных используется SPARQL, данные возвращаются в формате RDF/XML. DjangoRDF позволяет хранить данные в триплетах (triplestore), что обеспечивает работу с расширяемыми пользователем моделями данных. DjangoRDF включает в себя следующие компоненты:

транслятор языка запросов SPARQL в SQL, модели для хранения элементов онтологии, поддержка триплетов, вывод данных в RDF/XML, шаблоны и теги шаблонов. DangoRDF представляет модели из любого приложения Django в виде RDF-данных. Кроме этого, можно обращаться к хранилищу моделей в терминах классов и свойств RDFS или OWL, поддерживается RDF-хранилище с использованием внутренних сущностей, таких как концепция, предикат, ресурс, высказывание, литерал, онтология, пространство имен и т.д. Язык запросов SPARQL возвращает наборы запросов, которые позволяют свободно объединять данные из хранилища RDF с существующими моделями Django. Во время настройки DjangoRDF выполняется анализ схемы базы данных и создаются OWL онтологии для остальных установленных приложений. Онтологии сериализуются в виде файлов RDF/XML и последовательно десериализуются для создания комбинированного словаря в виде экземпляров модели RDF. Классы и свойства RDFS/OWL в словаре ассоциируются с элементами схемы базы данных, которые в последствии будут использоваться транслятором SPARQL для генерации SQL запросов. Реализация SPARQLQuerySet предоставляет доступ к экземплярам моделей, используя стандартный интерфейс наборов запросов Django: filter, count, iteration with slices и т. п. Важными недостатками данной платформы являются отсутствие поддержки локализации и неполная поддержка SPARQL. Кроме того, и это следует считать наиболее существенным недостатком, DjangoRDF в настоящее время не поддерживается и не развивается. В частности, невозможно использование DjangoRDF с последними версиями Django.

2.4.2 CubicWeb

Среда разработки семантических веб-приложений CubicWeb представляет собой семантический фреймворк, который предоставляет базовые типы данных, средства создания составных типов данных и отношений между объектами, инструменты описания схемы данных и способов их отображения. На основе описания схемы данных генерируется экземпляр приложения, а взаимодействие между пользователем и приложением осуществляется через web-интерфейс. Таким образом, для создания полнофункционального приложения разработчик должен описать модель предметной области и правила отображения объектов предметной области, необходимые экранные формы ге-

нерируются автоматически. [4]. Cubicweb определяет базовые типы данных для схемы, такие как String, Int, Float, Decimal, Boolean, Date, Datetime, Time, Interval, Byte, Password. Более сложные типы строятся на основе базовых. Cubicweb позволяет описывать так называемые кубы — схемы данных и способы их отображения. Необходимо создать экземпляры этого куба, чтобы инициировать заполнение базы данных и ввод данных пользователем. Экземпляр куба как правило запускает веб-сервер, что позволяет легко вводить и отображать данные посредством веб-браузера. В качестве недостатков данной среды стоит отметить невозможность последующей модификации созданной схемы данных, а также отсутствие средств для создания нестандартных веб-приложений.

2.4.3 Ontopia

Ontopia является программным пакетом с открытым исходным кодом для создания приложений на основе Topic Maps, обладает такими средствами, как редактор онтологий, редактор данных, полнофункциональный язык запросов, интеграция с web-сервисами, работа с различными базами данных в качестве хранилищ. Первая версия Ontopia 1.0 была выпущена в июне 2001 года, текущая версия Ontopia 5.1. Ontopia используется в ряде коммерческих проектов. Ontopia выпущена под лицензией Apache 2.0. Существует множество собственных и сторонних дополнений. Ontopia активно поддерживается и развивается участниками из различных стран: Германии, Нидерландов, Бельгии, Норвегии.

Ontopia включает в себя компоненты Omnigator, Ontopoly и Vizigator.

Omnigator используется для отладки созданной онтологии и является web-редактором Topic Maps. Он не является инструментом для конечных пользователей, а предназначен для использования разработчиками, поскольку он может быть использоваться для просмотра любой части любой онтологии, для проверки онтологии, получения статистики.

Ontopoly — редактор онтологий, позволяющий поэтапно создавать онтологию используя удобный веб-интерфейс. Пользователю также предоставляется веб-интерфейс для наполнения созданной онтологии данными. Созданную онтологию можно сохранить в виде файла Topic Map или в базе данных. Кроме того, Ontopoly может встраиваться в другие приложения, так что мож-

но добавить возможности редактирования Topic Maps в другие приложения, например, в системы управления контентом.

Компонент Vizigator служит для графического отображения созданной онтологии, визуализирует структуру Topic Maps, что может быть полезным для отображения больших моделей данных. Vizigator состоит из двух частей: VizDesktop, графический интерфейс для настройки визуализации, и Vizlet, апплет Java для отображения визуализации в web-браузере. Настройка визуализации не требует программирования, необходимо только сконфигурировать VizDesktop и развернуть java-апплет вместе с необходимым интерфейсом web-сервиса на стороне сервера.

В качестве хранилища данных Ontopia может использовать MySQL, PostgreSQL, Oracle, Microsoft SQL Server.

Для хранения семантической сети Ontopia использует формат Topic Maps, другие рассмотренные фреймворки используют для этих целей RDFS. Тем не менее, в рамках данной работы формат хранения данных семантической сети не является существенным, потому что работа идет с верхнеуровневым представлением данных.

В таблице 2.1. представлена сравнительная таблица функциональных возможностей различных семантичских фреймворков:

Таблица 2.1.

Сравнение семантических фреймворков

Критерий сравнения	DjangoRDF	CubicWeb	Ontopia
Редактор онтологий	-	+	+
Визуализация онтологии	-	-	+
Изменение схемы данных на лету	+	-	+
Актуальность состояния	-	+	+

После сравнительного анализа для разработки данной системы было принято решение использовать семантический web-фреймворк Ontopia.

3. РЕАЛИЗАЦИЯ СИСТЕМЫ

3.1 Технология создания учебных курсов

В качестве базовой идеи, которая легла в основу описываемого подхода, взята модель обучения во многих мировых университетах, при которой основу для изложения теоретического материала составляет некоторый набор учебников или монографий, на базе которого строятся сразу несколько курсов. Такие учебники обычно содержат больше информации, чем можно вложить в один курс, поэтому зачастую на основе одного и того же учебника строятся несколько разных курсов, каждый из которых покрывает свой аспект предметной области. В то же время учебник предлагает некоторую последовательность изложения материала, которая основана принципе использования только ранее определенных понятий. При этом часть понятий вводится в самом учебнике, а часть считается уже известной читателям. В зависимости от целей курса, возможностей аудитории и от собственных предпочтений, автор курса может в определенной степени менять порядок изложения материала, опускать некоторые темы или, напротив, расширять излагаемый материал, используя другие учебные пособия или результаты собственных исследований и методической работы. Тем не менее, в большинстве случаев курсы, автор которых опирается при построении курса на определенный учебник, в целом разделяют с этим учебником общую идеологию. Поскольку во многих случаях преподаватель часто читает несколько курсов, посвященных схожей тематике, в этих курсах он зачастую использует общее подмножество определений, формулировок, примеров, задач. В частности, большинство преподавателей регулярно обновляют содержание курса, сохраняя его базовые элементы. Чтобы сократить трудозатраты, связанные с подготовкой учебных курсов, а также упростить поиск и применение материалов курса студентами, представляется целесообразным строить систему управления курсами на основе формальизованных структуры курса и структуры предметной области, обеспечивая таким образом расширенные по сравнению с традиционными системами возможности поиска учебных материалов, их использования и управления ими.

Для создания учебных курсов с помощью описываемой системы управления учебными материалами предлагается следующая технология.

1. Формируется структура учебных материалов, соответствующая описываемой предметной области.
2. На основе структуры учебных материалов создается набор учебных материалов.
3. Формируется структура учебного курса.
4. На основе структуры курса и набора учебных материалов создается конкретный курс.
5. Формируется линейное представление курса с необходимыми гиперссылками.

При построении курсов преподаватели ориентируются на определенный набор учебников, монографий, статей или собственных результатов. Обычным требованием при разработке курса является наличие одного или нескольких современных учебников, используемых в качестве основного учебного пособия. Преподаватели выбирают или создают учебники таким образом, чтобы способ изложения материала в них соответствовал желаемому способу изложения материала в рамках курса. Таким образом, можно считать, что структура учебных материалов может быть основана на структуре учебника или набора учебников, используемых в данной предметной области.

- Построить базовую структуру учебных материалов предметной области в сотрудничестве с инженером по знаниям, выделяя ключевые элементы и связи, позволяющие наиболее полно описать выбранную предметную область.
- На основе сформированной структуры учебных материалов автор курса должен наполнить онтологию понятиями непосредственно из данной предметной области и установить связи между этими понятиями, указывая их тип и направление.
- Создать учебный курс, наполняя его объектами из полученной онтологии предметной области, и указать правила отображения этих объектов.

При описании структуры учебных материалов автор курса вносит в нее необходимые характерные для предметной области типы элементов: определения, теоремы, доказательства, утверждения, задачи, решения, алгоритмы, описания экспериментов. Элементы курса обычно связаны между собой, и для отражения этих связей автор добавляет в описание структуры учебных матери-

алов набор отношений: «использует», «доказывает», «иллюстрирует», «является решением». Ключевым является отношение использования: оно определяет логическую последовательность изложения материала, поэтому, в отличие от любых других отношений, отношение использования является обязательным в любой структуре учебных материалов.

Примеры расширения структуры учебных материалов для различных предметных областей приведены на рис. 4.1. (математика) и рис. 3.2. (химия).

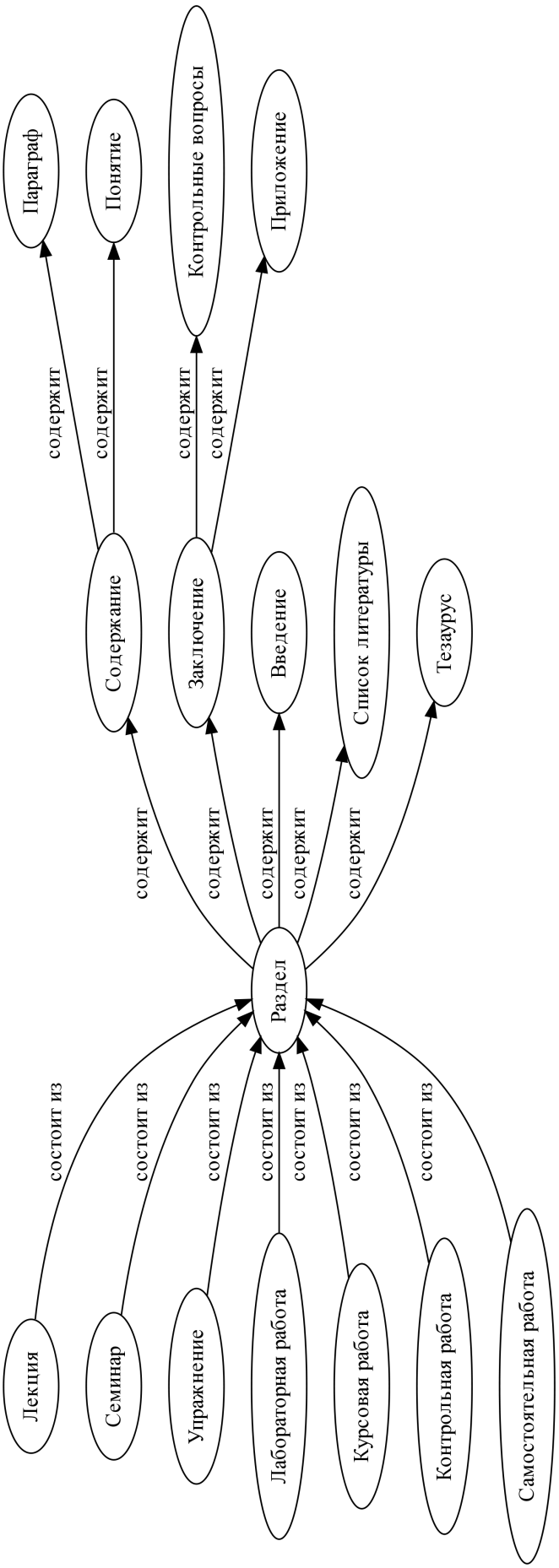


Рис. 3.1. **Базовая структурная схема курса**

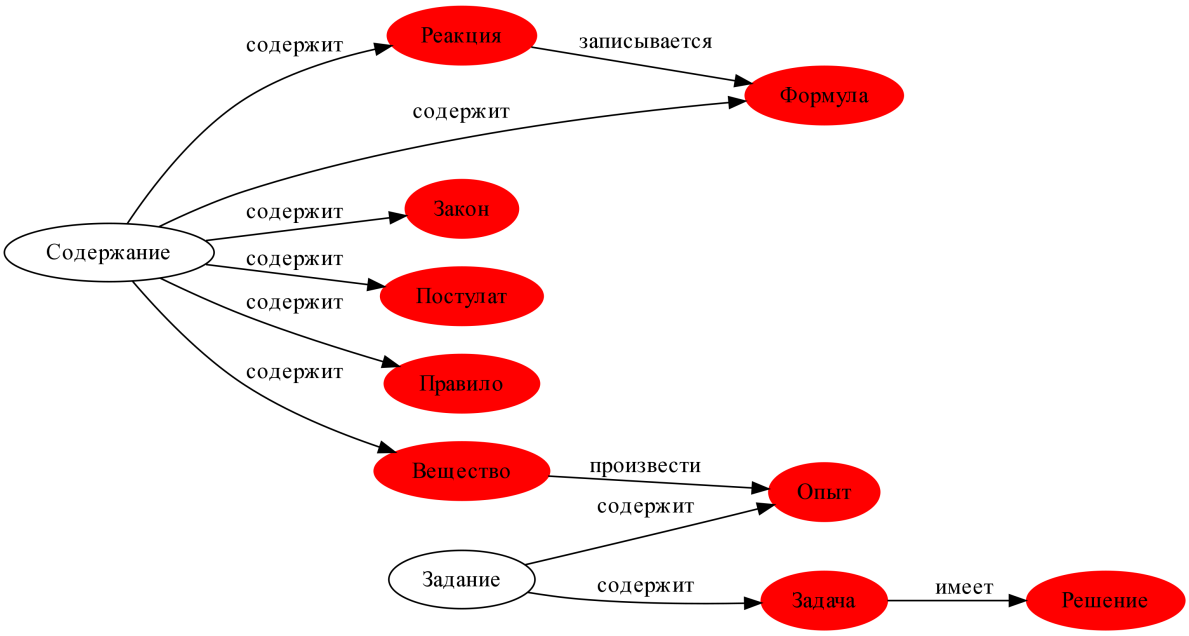


Рис. 3.2. **Структура учебных материалов предметной области «Химия»**

Наполнение набора учебных материалов представляет собой наиболее трудоемкий этап работы. На этом этапе автор курса непосредственно создает набор материалов, типы которых были определены на этапе формирования структуры учебных материалов. Описание каждого материала включает в себя наименование материала, его содержание и тип. Например, структура материала для школьного курса геометрии будет содержать элемент «теорема», а набор учебных материалов будет включать экземпляры этого типа: «теорема Пифагора», «теорема Фалеса», «теорема синусов». Если структура учебных материалов предполагает возможную связь (отношение) между некоторыми элементами курса, автор может сформировать такую связь между экземплярами учебных материалов. Например, отношение «использует» должно быть задано для пар («теорема Пифагора», «прямоугольный треугольник»), («теорема Пифагора», «квадрат числа»), («теорема Фалеса», «параллельные прямые»), при этом отношение использования может не включать пару («теорема Пифагора», «параллельные прямые»), поскольку формулировка теоремы Пифагора не опирается на понятие параллельных прямых.

Первый этап работы, формирование структуры учебных материалов, для каждой предметной области выполняется однократно. При необходимости, дальнейшие изменения могут выполняться инкрементально. Также однократно может быть выполнен и этап первоначального создания набора учебных

материалов. Эта работа может быть разделена между несколькими авторами, которые читают курсы на смежные темы. После того, как базовый набор учебных материалов будет создан, модификация этого набора уже не будет занимать много времени.

Этап формирования структуры учебного курса позволяет определить множество элементов, из которых строится учебный курс. В частности, был выделен следующий набор элементов:

- раздел;
- содержание;
- заключение;
- введение;
- список литературы;
- тезарус;
- контрольные вопросы;
- параграф;
- приложение;
- понятие;

Помимо перечисленных элементов, описание структуры курса должно содержать виды занятий, которые могут проводиться в рамках курса. Перечислим список таких занятий [18].

- семинар;
- лекция;
- упражнение;
- лабораторная работа;
- курсовая работа;
- контрольная работа;
- самостоятельная работа.

Перечисленные элементы входят в большую часть курсов, что позволяет считать их базовыми для описания структуры курса. При необходимости автор может добавить новые элементы структуры курса.

Между элементами структуры курса вводится набор отношений, описывающих, во-первых, вложенность структурных элементов (например, раздел

состоит из параграфов), и, во-вторых, возможность включения элементов курса в учебные занятия (понятие изучается на лекции).

Базовая структура учебного курса, представленная в виде онтологической модели, изображена на рис. 3.1.

Конкретный курс строится путем выбора необходимых элементов курса и привязывания к ним выбранных из сформированного набора учебных материалов. При этом автор курса сам определяет порядок изложения материала. На основе отношений между элементами курса, и отношений, определенных на множестве учебных материалов, можно провести формальную проверку логической корректности курса, в частности, проверку того, что все используемые понятия были определены в рамках курса. Кроме наполнения курса, автор может указать набор учебных материалов, таких, как определения или теоремы, знание которых студентом является предварительным требрванием для изучения курса. Линейное представление курса формируется автоматически на основе отношений предшествования на множестве элементов курса и на множестве учебных материалов.

3.2 Практическая реализация

На рис. 3.5. приведена общая структурная схема семантического вебфреймворка Ontopia, используемого при реализации пользовательского webприложения. Приложение написано на языке программирования Java с ипользованием механизма JavaServer Pages (JSP), используя API [24] и библиотеку, предоставляемую фреймворком Ontopia.

В рамках данной работы в качестве хранилища данных Ontopia будет использоваться реляционная база данных MySQL.

Для работы с хранилищем Topic Maps через программный интерфейс Ontopia используется интерфейс TopicMapIF, позволяющий считывать данные из онтологии, а также следующие программные интерфейсы:

- TopicMapSynchronizer — используется для синхронизации изменений внесенных в онтологию с хранилищем Topic Maps;
- TopicMapStoreIF — отвечает за установку соединения между выбранным Topic Map и хранилищем RDBMS;

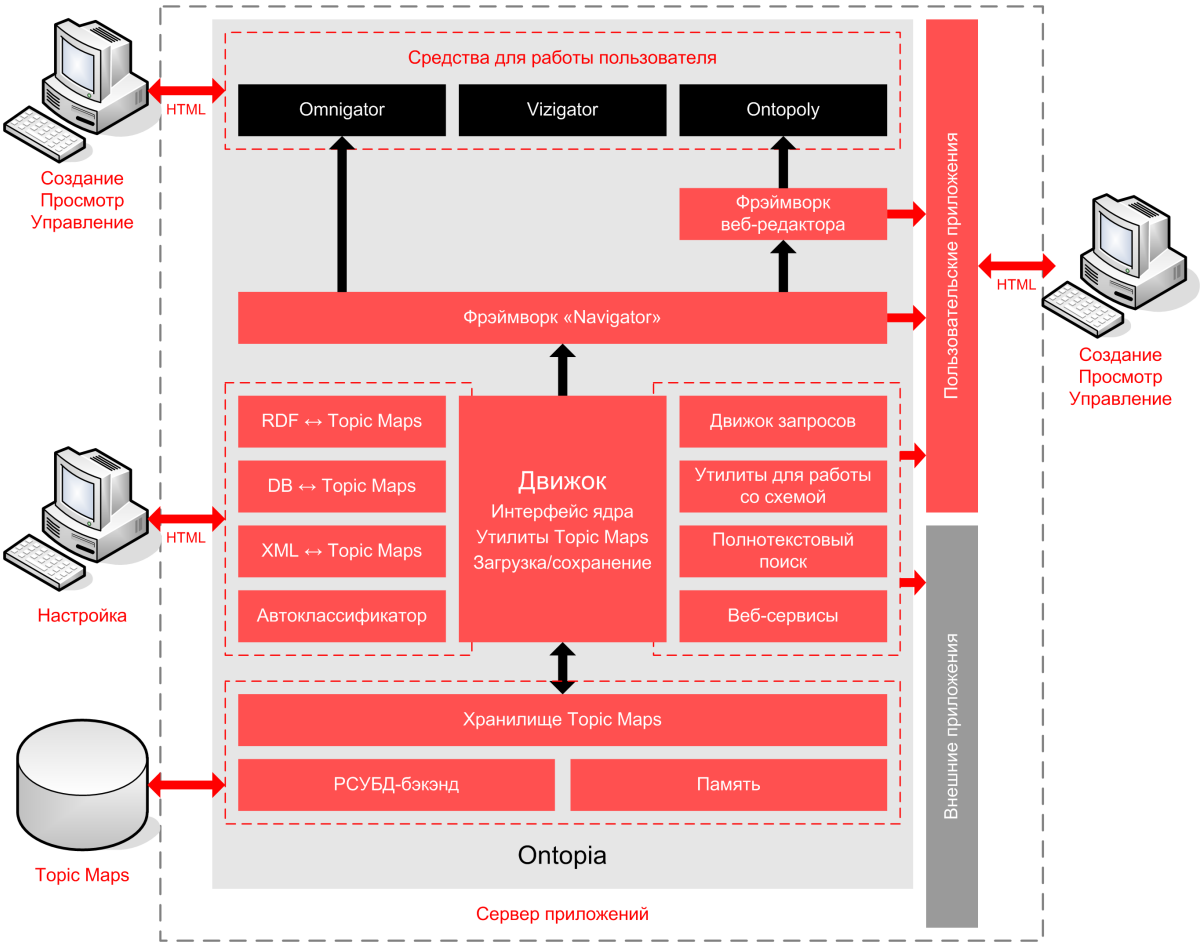


Рис. 3.3. **Структурная схема Ontopia**

– TopicMapWriterIF — записывает изменения внесенные в онтологию в схему данных;
– TopicIF — используется для получения сущности из выбранного Topic Map и связи между сущностями;
– TopicType — содержит в себе описание классов описывающих онтологию;
– FieldsView — позволяет выбрать все свойства выбранного объекта.

Для просмотра полученной онтологии предметной области используется библиотека Graphviz Java [21], позволяющий получить графическое отображение онтологии, полученной из хранилища Topic Map. Архитектура клиент-серверного приложения приведена на рис. 3.6..

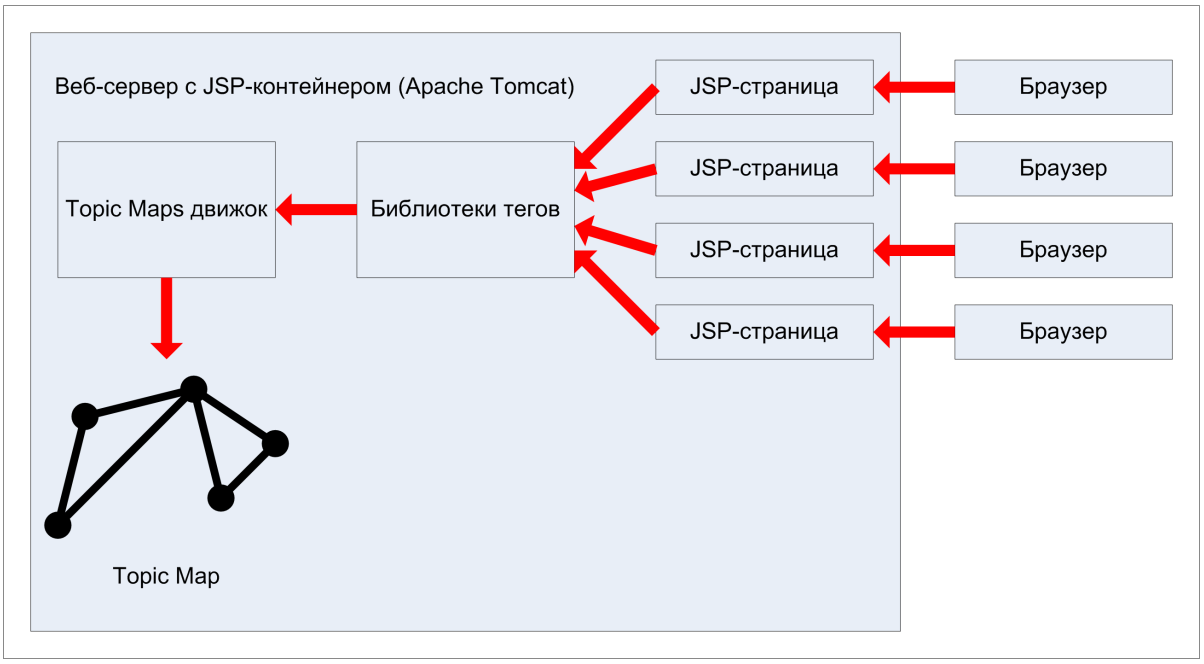


Рис. 3.4. **Схема работы приложения**

3.2.1 Архитектура системы

На рис. 3.5. приведена общая структурная схема семантического веб-фреймворка Ontopia, используемого при реализации пользовательского веб-приложения. Пользовательское приложение написано на языке программирования java с ипользованием механизма JavaServer Pages (JSP), через API [24] и библиотеку предоставляемую фреймворком Ontopia.

В рамках данной работы для хранилища данных в инструменте Ontopia будет использоваться реляционная база данных MySQL.

Для работы с хранилищем Topic Maps через программный интерфейс Ontopia используется интерфейс TopicMapIF, позволяющий считать данные из онтологии, а также следующие программные интерфейсы:

- TopicMapSynchronizer — используется для синхронизации изменений внесенных в онтологию с хранилищем Topic Maps;
- TopicMapStoreIF — отвечает за установку соединения между выбранным Topic Map и хранилищем RDBMS;
- TopicMapWriterIF — записывает изменения внесенные в онтологию в схему данных;
- TopicIF — используется для получения сущности из выбранного Topic Map и связи между сущностями;

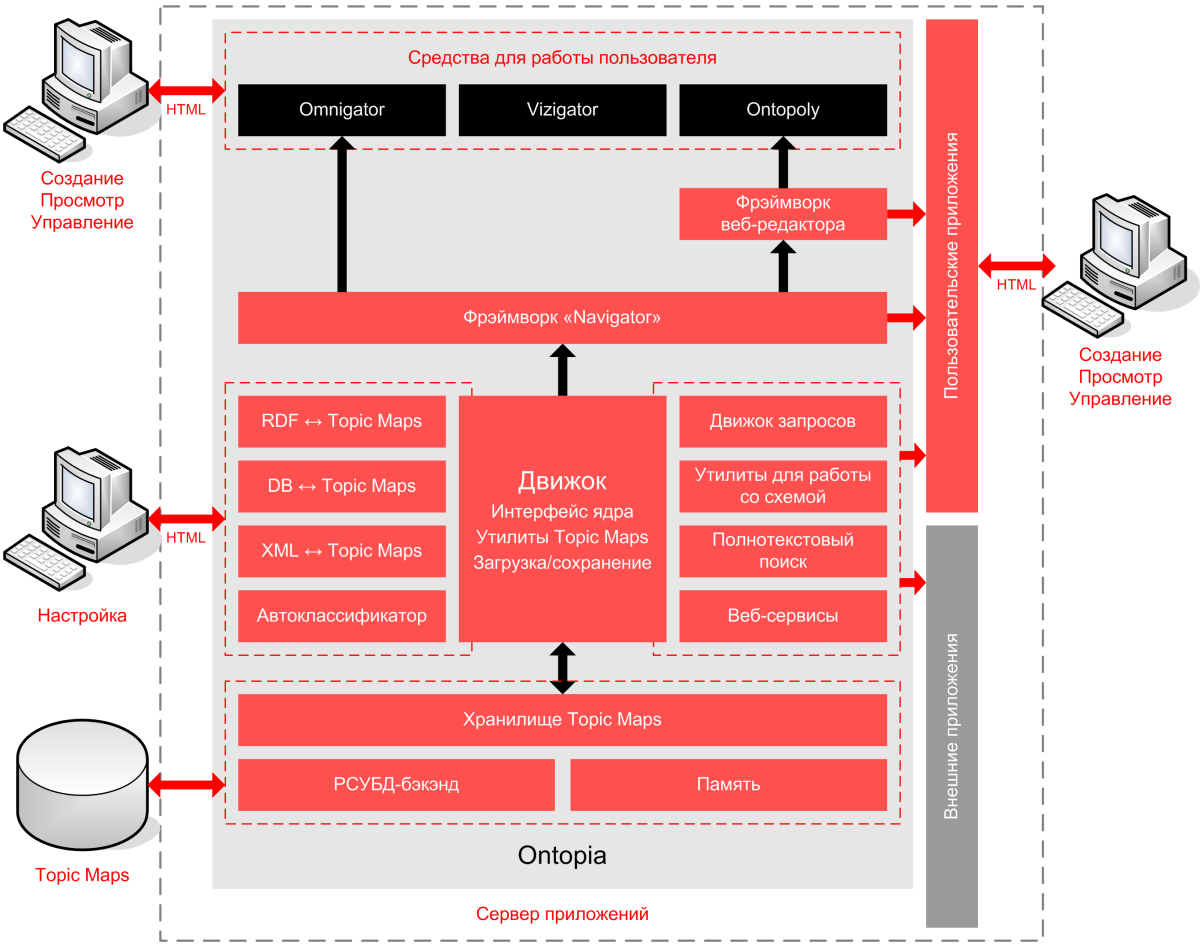


Рис. 3.5. **Структурная схема Ontopia**

– TopicType — содержит в себе описание классов описывающих онтологию;
– FieldsView — позволяет выбрать все свойства выбранного объекта.

Для просмотра полученной онтологии предметной области используется плагин Graphviz Java [21], позволяющий получить графическое отображение онтологии полученной из хранилища Topic Map. Архитектура клиент-серверного приложения приведена на рис. 3.6.

3.2.2 Интерфейс прототипа системы

В разработанной системе были реализованы следующие основные формы:

– главная страница;
– просмотр онтологии предметной области;
– редактирование онтологии предметной области;

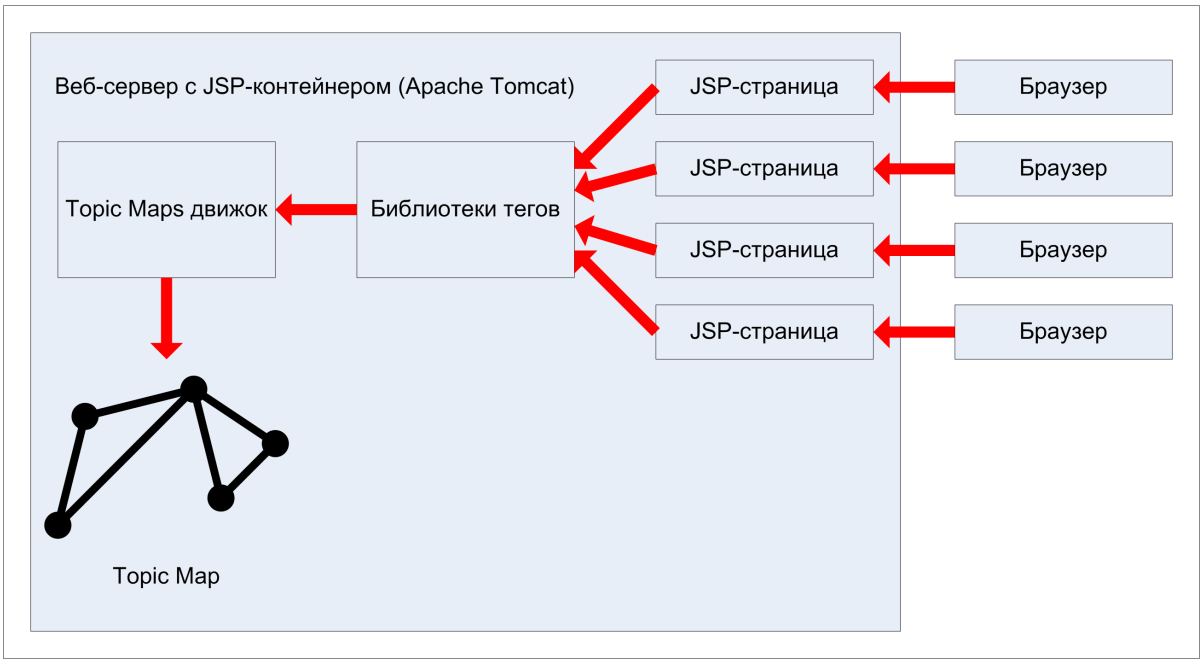


Рис. 3.6. **Схема работы приложения**

– просмотр имеющихся учебных курсов;
– добавление нового учебного курса;
– редактирование выбранного учебного курса.

Система не предоставляет пользователю интерфейса для работы со структурой учебных материалов предметной области, поскольку такая работа должна быть выполнена однократно с привлечением инженера по знаниям в данной предметной области. Эта операция осуществляется через веб-интерфейс Ontopoly семантического веб-фреймворка Ontopia. При необходимости внесения изменений в созданное описание также потребуется привлечение инженера по знаниям, с которым необходимо согласовать все предлагаемые исправления и дополнения.

Для начала работы на главной странице необходимо выбрать требуемый предмет с помощью ниспадающего списка, после чего можно осуществлять переход на другие формы.

Форма просмотра онтологии предметной области позволяет отобразить в графическом виде имена объектов и связи между ними. Построение изображения на основе хранящейся в системе онтологии осуществляется посредством дополнения [21]. Пример отображаемой формы просмотра онтологии предметной области показан на рис. 3.7.

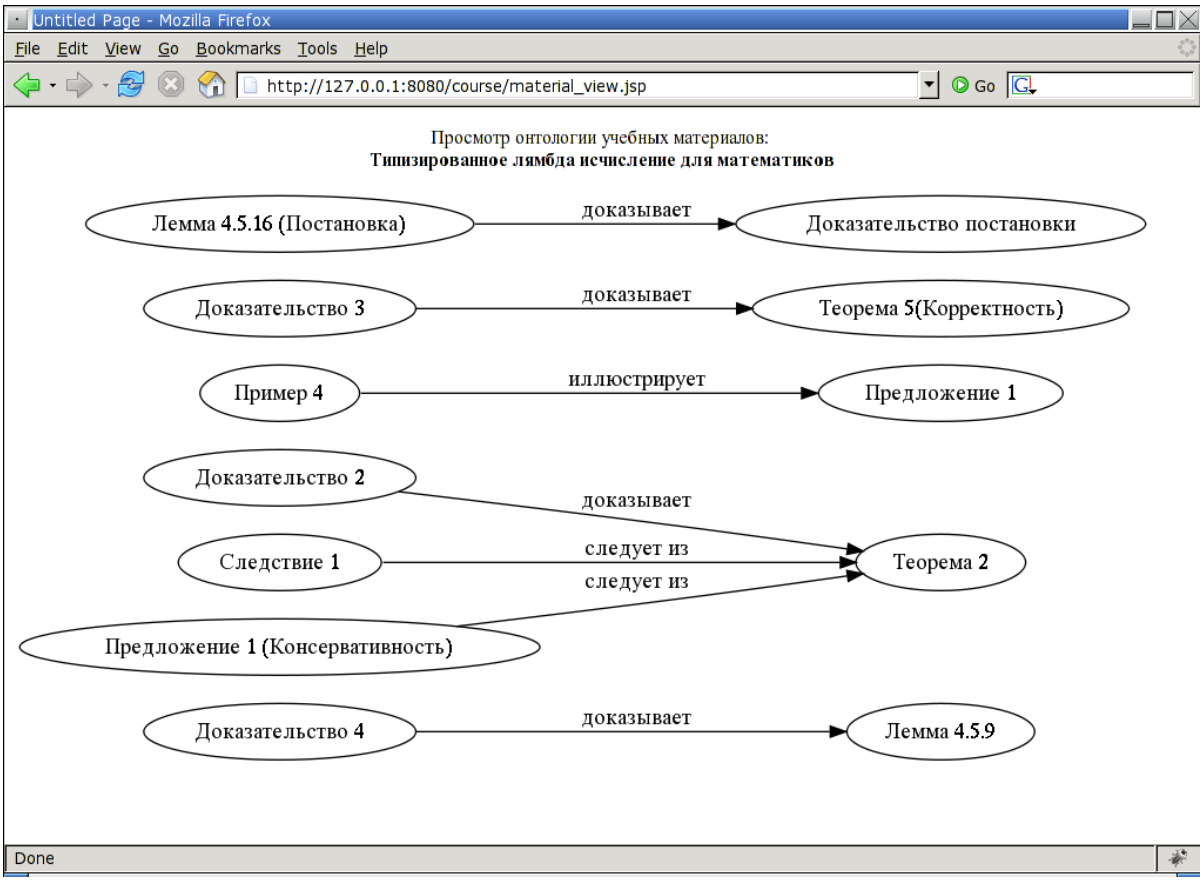


Рис. 3.7. **Просмотр онтологии предметной области**

Наполнение и изменение онтологии выбранной предметной области осуществляется с использованием формы редактирования онтологии. Элементы управления данной формы позволяют добавлять и удалять объекты, устанавливать связи выбранного типа между заданными объектами, просматривать список имеющихся объектов с возможностью отображения только объектов определенного типа, просматривать и изменять значение выбранного объекта, а также его имя, просматривать связи выбранного объекта с другими объектами и удалять ошибочно установленные связи. Для добавления нового объекта необходимо указать его имя в текстовом поле (например, «Теорема Пифагора») и нажать кнопку «Добавить объект», после чего в открывшейся форме добавления объекта нужно указать тип и ввести значение объекта, например, для теоремы — ее формулировку. После нажатия кнопки «Сохранить» объект выбранного типа с указанными именем и значением будет добавлен в онтологию предметной области. Чтобы установить новую связь между объектами онтологии, необходимо выбрать тип связи в ниспадающем списке и нажать кнопку «Добавить связь». В результате откроется форма связывания объектов, где по-

требуется выбрать из ниспадающих списков связываемые объекты и нажать кнопку «Сохранить». Связь между выбранными объектами устанавливается в следующей последовательности: субъект — предикат — объект. Например, «Доказательство методом площадей» «доказывает» «Теорема Пифагора». В нижней части формы редактирования онтологии предметной области отображается список объектов, ранее добавленных в онтологию, и ниспадающий список фильтра класса, который позволяет просматривать не только весь список, но и, например, выводить только объекты класса «теорема» или «следствие». Пример отображаемой формы редактирования онтологии предметной области показан на рис. 3.8..

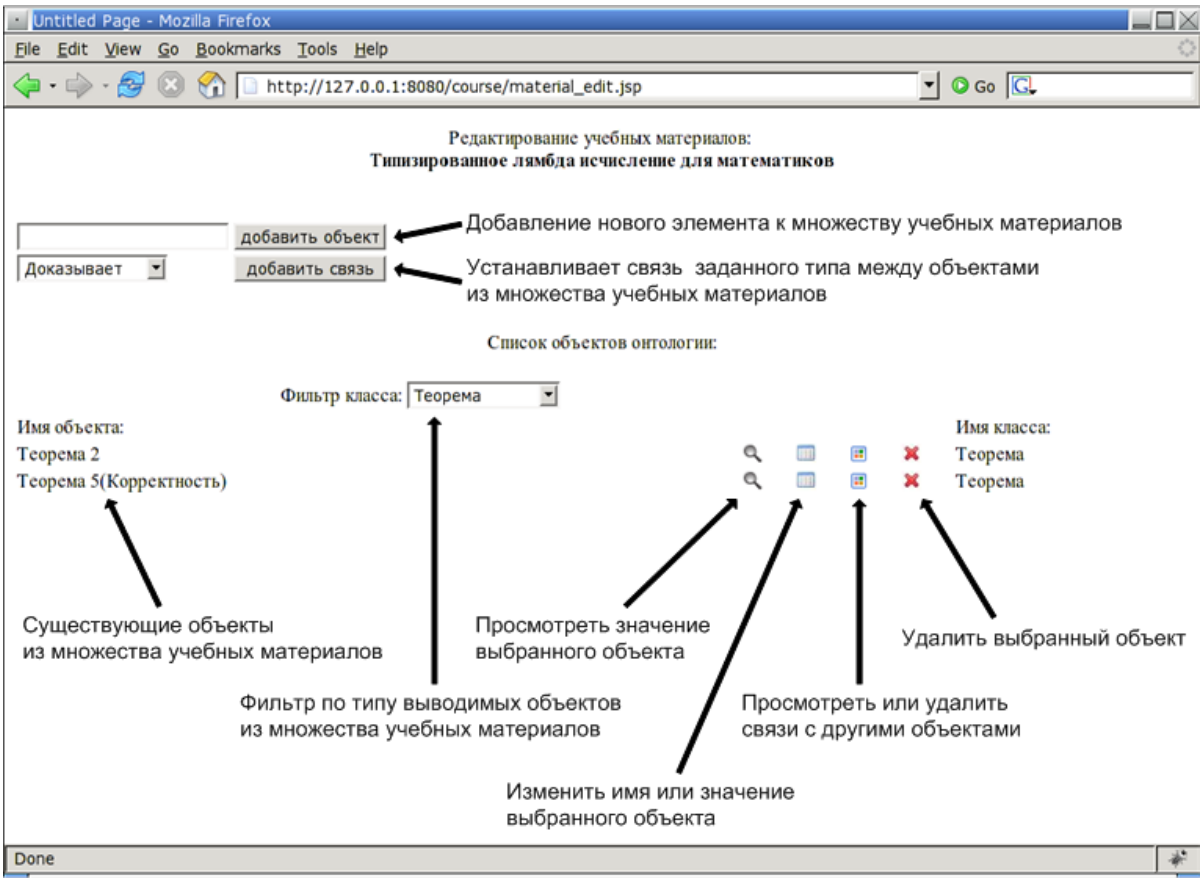


Рис. 3.8. **Редактирование онтологии предметной области**

Просмотр значения определенного объекта осуществляется посредством формы просмотра объекта, вызываемой нажатием на иконку просмотра в списке напротив выбранного объекта. На данной форме отображаются имя объекта, его класс и описательное значение.

Нажатие на иконку редактирования в списке напротив выбранного объекта открывает форму редактирования объекта, где можно выбрать тип объекта

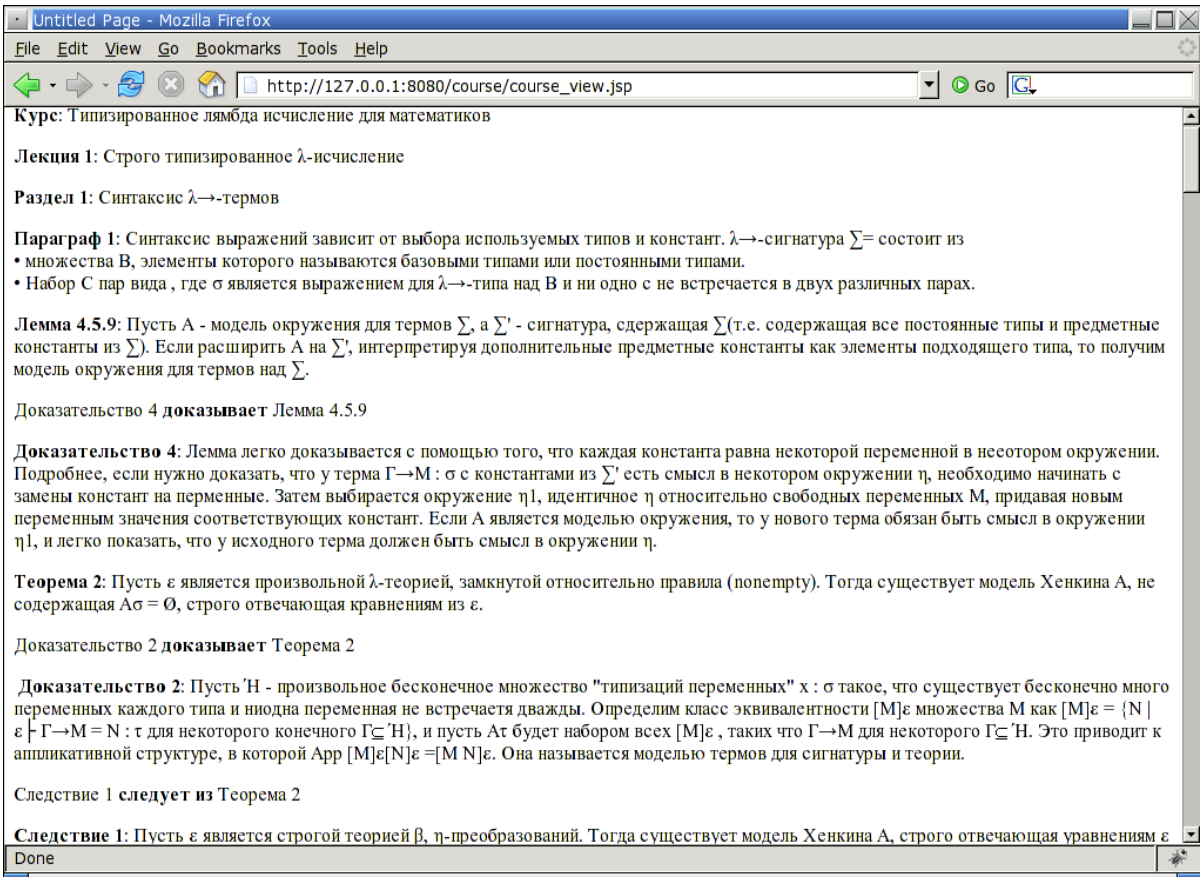
из ниспадающего списка, а также изменить его имя и значение, используя соответствующие текстовые поля. После нажатия кнопки «Сохранить» изменения будут сохранены в онтологии.

Просмотреть все связи определенного объекта и удалить ошибочно установленные связи можно на форме просмотра связей объекта, которая открывается после нажатия на иконку просмотра связей в списке напротив требуемого объекта. На форме будет отображен список, в котором указано имя выбранного объекта, тип связи и имя объекта, связанного с заданным. Также напротив каждого пункта списка имеется иконка удаления связи, нажатие на которую открывает диалоговое окно с запросом на подтверждение удаления связи. После подтверждения выбранная связь между указанными в списке объектами будет удалена.

Удаление выбранного объекта производится нажатием на иконку удаления объекта в списке напротив объекта, который требуется удалить. В результате будет открыто диалоговое окно с запросом на подтверждение удаления объекта. После подтверждения выбранный объект будет удален из онтологии.

Форма просмотра учебных курсов позволяет увидеть список с названиями всех созданных учебных курсов выбранной предметной области. В верхней части формы расположен ниспадающий список, в котором можно выбрать определенного автора курса, чтобы просмотреть только те учебные курсы, которые были составлены с участием выбранного автора. По нажатию на иконку просмотра курса в списке напротив названия выбранного курса откроется форма просмотра учебного курса. На этой форме отображается название курса, перечисляются его авторы, и в порядке, заданном составителем курса, отображается весь входящий в данный учебный курс материал: объекты с именами и значениями, связанные с ними другие объекты с указанием их имен и значений, а также типы связей между этими объектами. Пример отображаемой формы просмотра учебного курса показан на рис. 3.9.

Добавление нового учебного курса осуществляется с помощью формы добавления курса, на которой необходимо указать название курса и имя его автора, после чего необходимо нажать кнопку «Добавить автора». При необходимости указать имена нескольких авторов необходимо поочередно вводить имя каждого из них и нажимать кнопку «Добавить автора» По завершении этих операций нужно нажать кнопку «Сохранить», после чего в систему будет



Курс: Типизированное лямбда исчисление для математиков

Лекция 1: Строго типизированное λ-исчисление

Раздел 1: Синтаксис λ→-термов

Параграф 1: Синтаксис выражений зависит от выбора используемых типов и констант. λ→-сигнатура ∑= состоит из
• множества B, элементы которого называются базовыми типами или постоянными типами.
• Набор C пар вида , где σ является выражением для λ→-типа над B и ни одно с не встречается в двух различных парах.

Лемма 4.5.9: Пусть A - модель окружения для термов ∑, а ∑' - сигнатура, сдержащая ∑(т.е. содержащая все постоянные типы и предметные константы из ∑). Если расширить A на ∑', интерпретируя дополнительные предметные константы как элементы подходящего типа, то получим модель окружения для термов над ∑.

Доказательство 4 **доказывает** Лемма 4.5.9

Доказательство 4: Лемма легко доказывается с помощью того, что каждая константа равна некоторой переменной в нееотором окружении. Подробнее, если нужно доказать, что у терма Γ→M : σ с константами из ∑' есть смысл в некотором окружении η, необходимо начинать с замены констант на перменные. Затем выбирается окружение η1, идентичное η относительно свободных переменных M, придавая новым переменным значения соответствующих констант. Если A является моделью окружения, то у нового терма обязан быть смысл в окружении η1, и легко показать, что у исходного терма должен быть смысл в окружении η.

Теорема 2: Пусть ε является произвольной λ-теорией, замкнутой относительно правила (nonempty). Тогда существует модель Хенкина A, не содержащая Aσ = Ø, строго отвечающая кравнениям из ε.

Доказательство 2 **доказывает** Теорема 2

Доказательство 2: Пусть Ή - произвольное бесконечное множество "типизаций переменных" x : σ такое, что существует бесконечно много переменных каждого типа и ниодна переменная не встречаетя дважды. Определим класс эквивалентности [M]ε множества M как [M]ε = {N | ε ⊢ Γ→M = N : τ для некоторого конечного Γ⊆Ή}, и пусть Aτ будет набором всех [M]ε , таких что Γ→M для некоторого Γ⊆Ή. Это приводит к аппликативной структуре, в которой App [M]ε[N]ε =[M N]ε. Она называется моделью термов для сигнатуры и теории.

Следствие 1 **следует из** Теорема 2

Следствие 1: Пусть ε является строгой теорией β, η-преобразований. Тогда существует модель Хенкина A, строго отвечающая уравнениям ε

Рис. 3.9. **Просмотр учебного курса**

добавлен учебный курс без какого-либо содержимого. Для наполнения курса материалом необходимо использовать форму редактирования курса.

Форма редактирования учебного курса используется для добавления содержимого в учебный курс. Чтобы указать курс, в который будет добавляться содержимое, необходимо выбрать его название из ниспадающего списка в верней части формы и нажать кнопку «Выбрать». При добавлении содержимого нужно выбрать элемент структуры курса из соответствующего ниспадающего списка, затем выбрать объект из множества учебных материалов, для некоторых типов объектов (например, «параграф») потребуется заполнить текстовые поля «Имя» и «Значение», для других типов (например, «теорема») эти поля будут заполнены автоматически и станут недоступны для редактирования, поскольку эти объекты уже были заданы при создании онтологии предметной области и могут быть изменены только из формы редактирования онтологии. После этого нужно нажать кнопку «Добавить», в результате чего выбранный объект будет связан с элементом структуры курса. Добавление содержимого необходимо производить в том порядке, в котором это содержимое должно

отображаться при просмотре курса. В нижней части формы расположен список зависимых понятий, который заполнен такими объектами, которые имеют связь с добавляемыми в учебный курс. При необходимости составитель может включить эти связанные объекты в свой курс, установив напротив выбранных объектов флажки в графе «Отображение», после чего необходимо нажать кнопку «Сохранить отображение». Пример отображаемой формы редактирования учебного курса показан на рис. 3.10.

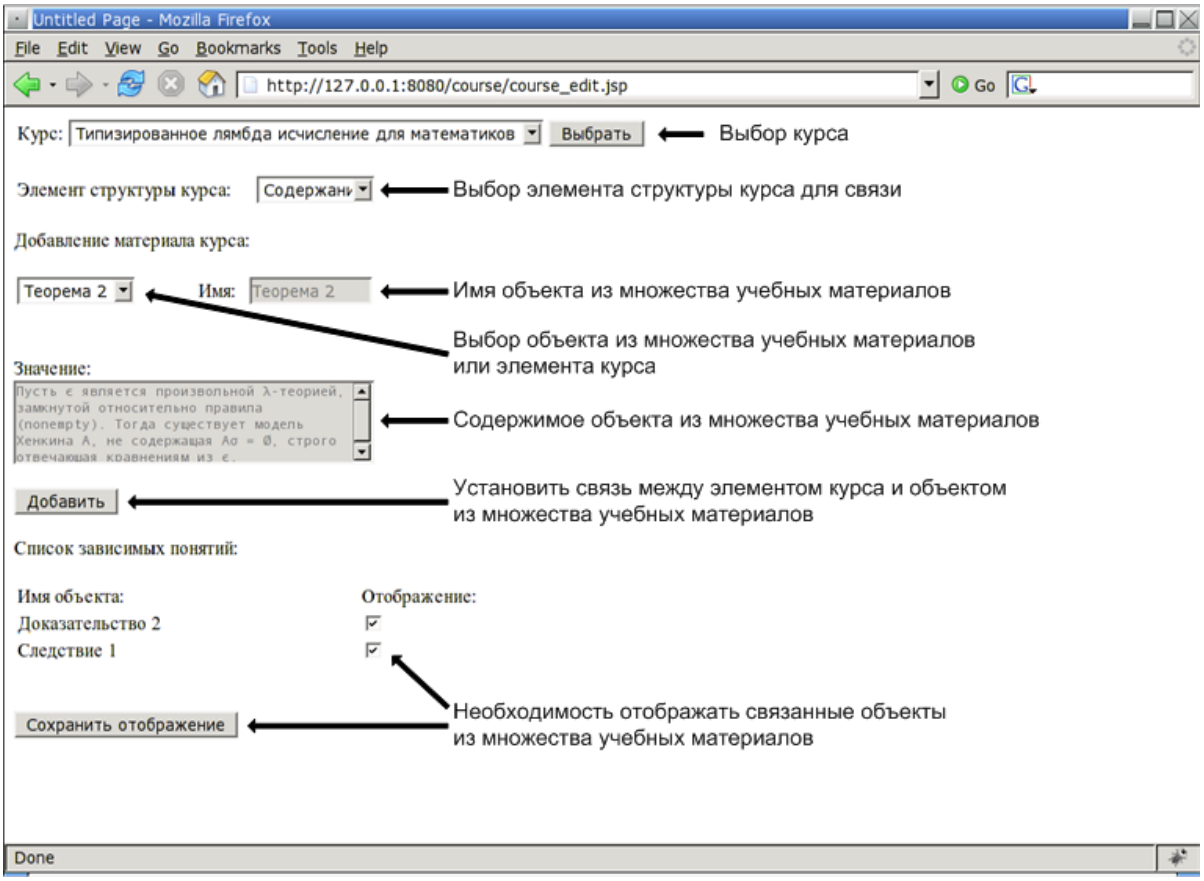


Рис. 3.10. **Редактирование учебного курса**

Общий граф связей между формами системы приведен на рис. 3.11.

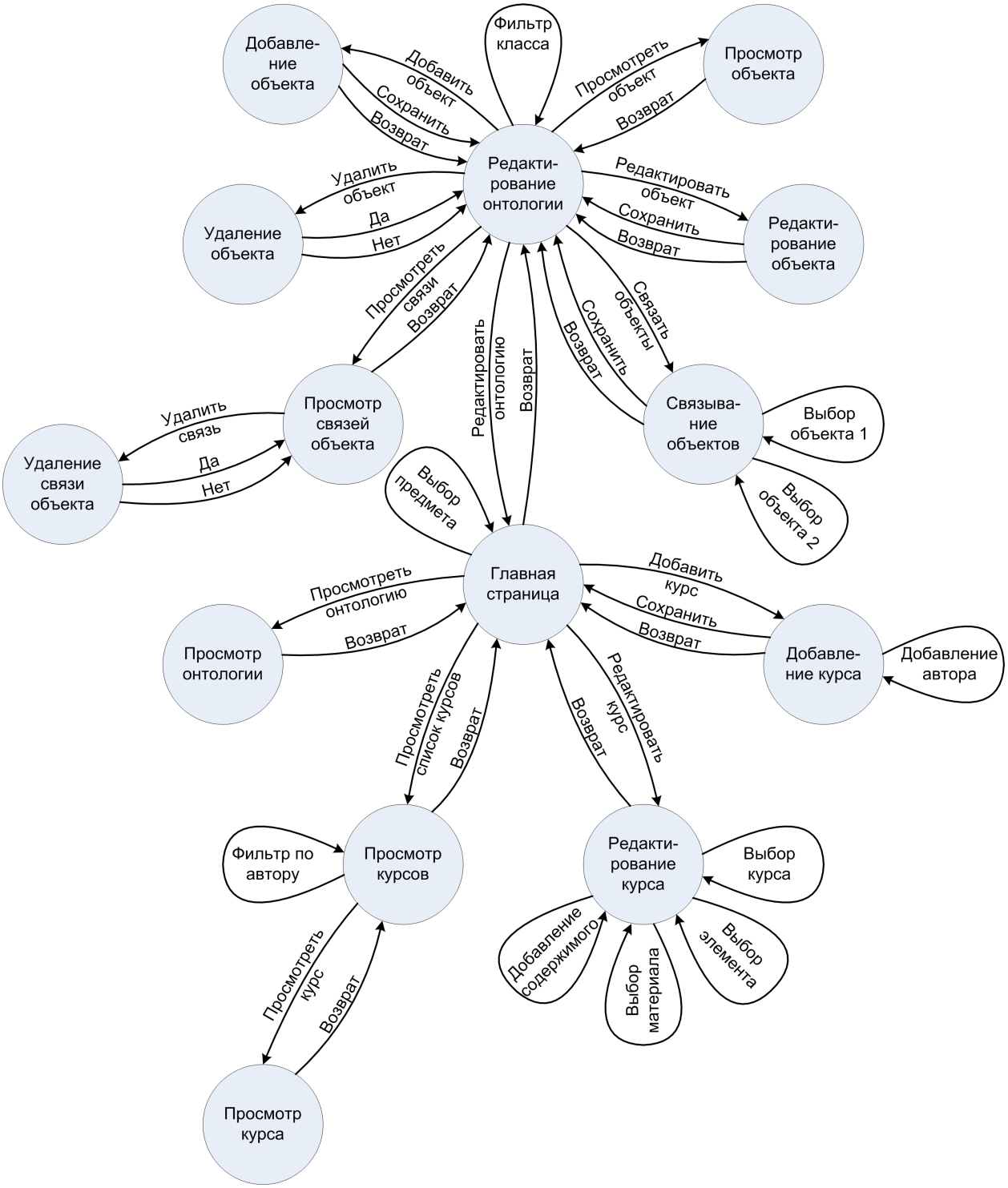


Рис. 3.11. **Граф связей системы**

4. ИСПЫТАНИЯ

4.1 Методика и условия проведения испытаний

В качестве методики испытаний разработанной системы предложена следующая последовательность:

1. На основе имеющейся онтологии структуры учебных материалов предметной области “формальная семантика языков программирования” разработать и добавить в систему онтологию предметной области.
2. Составить онтологию учебного курса по предмету, опираясь на созданную онтологию предметной области.
3. Составить программу курса на основе соответствующей онтологии.
4. Наполнить курс учебным материалом, списком рекомендуемой литературы, дополнительными материалами.
5. Основываясь на созданном курсе подготовить новый учебный курс, используя частично структуру и учебные материалы базового курса.

Для тестирования предложенной методики при работе с системой было решено взять части курсов приведенных на стр. 58, и составить по ним необходимый набор терминов для структуры описания учебных материалов соответствующая предметной области.

Набор понятий для предметной области связанной с математикой:

- теорема;
- лемма;
- аксиома;
- гипотеза;
- доказательство;
- задача;
- предложение;
- определение.

Связи между данными терминами отображены на рис. 4.1.

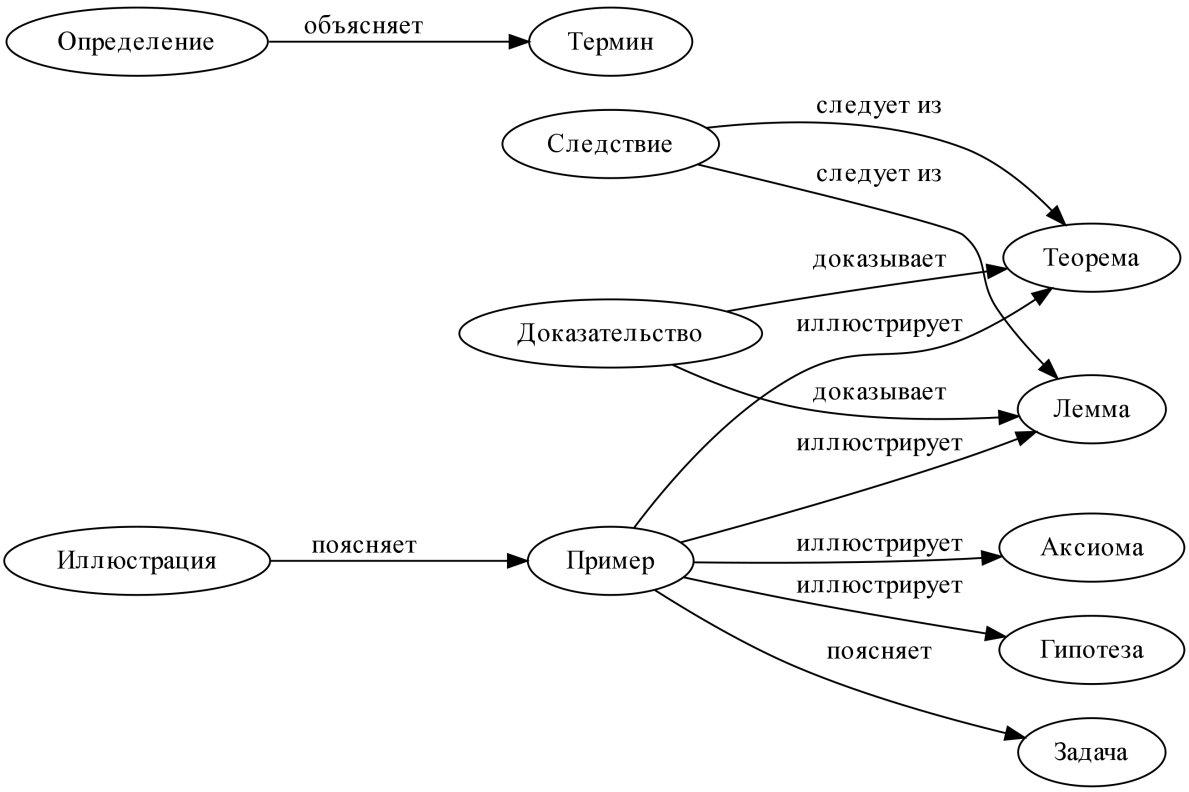


Рис. 4.1. Структура учебных материалов соответствующая предметной области “Математика”

4.2 Тестовые данные

Примеры курсов по книге [1].

Первый — вступительный курс, представляет из себя базовый курс по теории алгоритмов и автоматов и математической логике, без углубления в аспекты полноты, непротиворечивости и теорию моделей. Курс ориентирован на студентов математических специальностей, связанных с системным анализом. Желательным при изучении курса может быть включение главы по алгебраическим системам записи, приведение доказательств полноты, непротиворечивости и других свойств типизированного лямбда-исчисления или выборочное рассмотрение глав 9-11. Глава 3 может быть избыточна, если студенты уже знакомы с математической логикой.

Второй курс — требует большей математической подготовки. Он посвящен типизированному лямбда-исчислению и семантикам, содержит больше технических деталей.

Третий курс — включает системы типов, начиная с типизированного лямбда-исчисления и кончая полиморфизмом, подтипами и выводом типов.

Схема общего вводного курса:

1. Функциональное программирование и типизированное лямбда-исчисление (глава 2).
    1) Булевые значения, натуральные числа, пары и функции. Определение рекурсивных функций через оператор неподвижной точки (раздел 2.2).
    2) Сравнение аксиоматической, операциональной и денотационной семантики (раздел 2.3).
    3) Свойства редукции; детерминированная символьная интерпретация (раздел 2.4).
    4) Технологии программирования выразительная мощь и ограничения (раздел 2.5).
2. Универсальная алгебра и алгебраические типа данных (глава 3).
    1) Алгебраические термы и уравнения. Алгебры (разделы 3.1-4).
    2) Система вывода с равенством, полнота и корректность (раздел 3.4).
    3) Гомоморфизмы и инициальные алгебры (раздел 3.5).
    4) Элементы алгебраической теории типов данных (раздел 3.6).
3. Семантика типизированного лямбда-исчисления и рекурсия (главы 4 и 5).
    1) Представление контекстно-зависимого синтаксиса через правила типизации (разделы 4.3.1, 4.3.2 и 4.3.5).
    2) Общие модели, обзор полноты и корректности (разделы 4.5.1-4).
    3) Модели типизированного лямбда-исчисления с оператором неподвижной точки на областях (разделы 4.5.1-4).
4. Императивные программы (глава 6).
    1) Синтаксис while-программ; L-значения и R-значения (раздел 6.2).
    2) Операциональная структурная семантика (раздел 6.3).
    3) Денотационная семантика средствами типизированного лямбда-исчисления с оператором неподижной точки, типами location и store (раздел 6.4).

4) Частичная корректность. Корректность, относительная полнота и примеры доказательств (раздел 6.5).

Курс типизированного лямбда-исчисления для математиков:

1. Синтаксис и система вывода типизированного лямбда-исчисления.
   1) Контекстно-зависимый синтаксис и алгоритм типизации (разделы 4.1-3).
   2) Редурция и система вывода с равенством (раздел 4.4).
   3) Рекурсия с использованием оператора неподвижной точки (разделы 2.2.2-4, 2.2.5).
   4) Рекурсивные типы и точный подъем (раздел 2.6).
2. Теория моделей типизированного лямбда-исчисления.
   1) Общие определения, корректность и полнота (разделы 4.4.1, 4.5.1-6).
   2) Области (разделы 5.1, 5.2).
   3) Нумерованные множества (разделы 5.5, 5.6).
3. Логические отношения.
   1) Определения и базовые леммы (разделы 8.1, 8.2).
   2) Теоремы теории доказательств: полнота, сходимость, нормализация (раздел 8.3).
   3) Теоремы полноты для иерархии множеств, нумерованных множеств и областей (раздел 8.4).
4. Теория категорий и рекурсивные типы.
   1) Категории, функторы и естественные преобразования (разделы 7.1, 7.2.1-2).
   2) Декартово замкнутые категории и типизированное лямбда-исчисление (разделы 7.2.3-6).
   3) Пример категории, не являющейся вполне точечной. Лямбда-модели Крипке (раздел 7.3).
   4) Модели областей рекурсивных типов (раздел 7.4).

Курс теории типов:

1. Строго типизированное лямбда-исчисление.

1) Контекстно-зависимый синтаксис и алгоритмы типизации (разделы 4.1-3).
2) Редукция и система вывода с равенством (разделы 4.4.1, 4.4.2).

2. Полиморфизм.
   1) Введение в полиморфные типы (раздел 9.1).
   2) Предикативный полиморфизм (разделы 9.3.1-4).
   3) Абстрактные типы данных и объемные типы (раздел 9.4).
   4) Обобщенные произведения, суммы и программные модули (раздел 9.5).
3. Подтипы.
   1) Основной синтаксис, вложенность, равенство. Интерпретация подтипов через преобразования (разделы 10.1-10.4).
   2) Записи, рекурсивные типы, модель объектов через записи (раздел 10.5).
   3) Полиморфизм с условиями на подтипы (раздел 10.6).
4. Вывод типов.
   1) Вывод типов и стирающие функции (раздел 11.1).
   2) Вывод типов для строго типизированного лямбда-исчисления с помощью унификации (раздел 11.2).
   3) Полиморфные объявления в ML.

Проведя анализ Главы 4 из книги [1] были сформированы следующие понятия для структуры учебных материалов соответствующих предметной области “формальная семантика языков программирования”:

- теорема — утверждение, для которого в рассматриваемой теории существует доказательство;
- лемма — доказанное утверждение, полезное не само по себе, а для доказательства других утверждений;
- гипотеза — недоказанное утверждение, предположение или догадка;
- аксиома — утверждение, принимаемое истинным без доказательств;
- факт — знание в форме утверждения, достоверность которого строго установлена;

- пример — рассматривается в риторике чаще всего в контексте доказательств и аргументов;
- опровержение — рассуждение, направленное против тезиса с целью установления факта его ложности или недоказанности;
- доказательство — рассуждение по определенным правилам, обосновывающее какое-либо утверждение;
- утверждение — особая форма предложения, которая в утвердительной форме выдвигает гипотезу относительно некоторого явления;
- определение — формулировка, разъясняющая содержание, смысл чего-либо.

Связи между данными терминами отображены на рис. 4.1. Полная онтология курса «Математика» отображена на рис. 4.2.

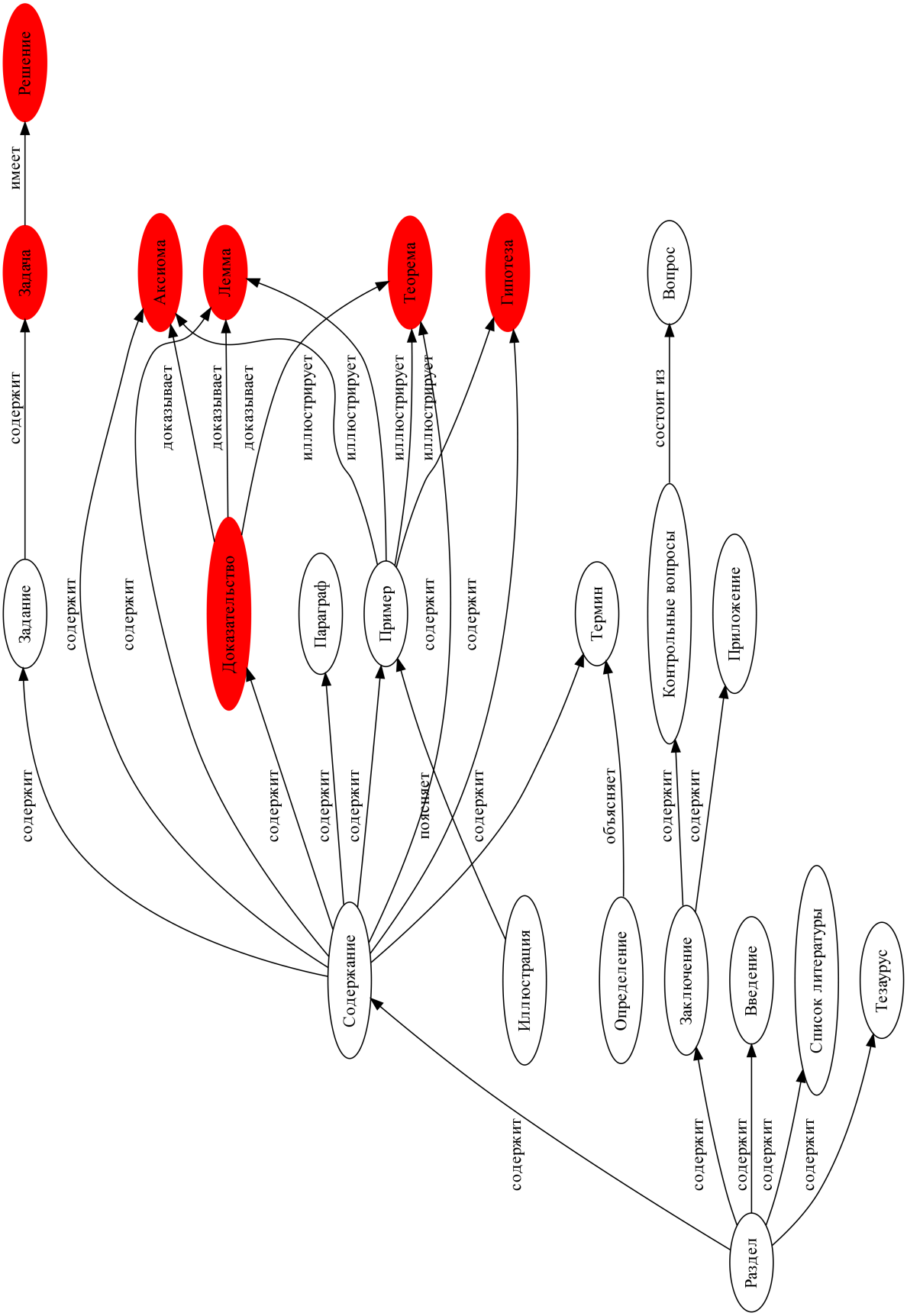


Рис. 4.2. Онтология курса “Математика”

4.3 Анализ результатов

В результате по предложенной методике было реализовано отображение части курсов описанных в главе 4.2 использующих смежные понятия. Результат отображения курса см. на рис. 3.9. Данная система позволяет выводить зависимости между понятиями использованными в курсе. Время ввода понятий в существующую систему сопоставимо с временем их ввода в системы LCMS, но при повторном использовании этих же понятий в других курсах время на добавление не тратится. Сводная таблица сравнительных возможностей полученного компонента с аналогами приведена в таблице 4.1.

Таблица 4.1.

Сравнение полученной LCMS

Критерий сравнения	Moodle/Sakai	Wiki	Разработанная LCMS
Типизированные связи между понятиями	—	—	+
Повторное использование материала в нескольких курсах	—	+	+
Добавление новых классов не требует изменения схемы БД	—	+	+
Обязательный порядок изучения материала	+	—	+
Частичный порядок изучения материала	—	—	+
Проверка отсутствия противоречий	—	—	+

ЗАКЛЮЧЕНИЕ

В результате выполнения работы:

- Разработана базовая структура учебных материалов, на основе которой построено описание одной предметной области — формальной семантики языков программирования. За основу описания взята монография [1].
- Построена базовая онтологическая модель учебного курса. На основе этой модели описаны два учебных курса, посвященных описанию различных фрагментов предметной области. Курсы отличаются друг от друга целью и базовым уровнем подготовки.
- Реализована система, поддерживающая управление учебными материалами на основе предложенного подхода.

Разработанная система включает, в частности, следующие модули:

- Модуль редактирования структуры учебных материалов.
- Модуль наполнения набора учебных материалов в соответствии с разработанной структурой.
- Модуль редактирования структуры учебного курса.
- Модуль создания учебного курса на основе структуры курса и набора учебных материалов.
- Модуль отображения готового курса для учащихся.

В качестве возможных направлений дальнейшего расширения функциональности системы можно выделить поддержку стандарта LOM (см. раздел 1.2.4), а также реализацию многоуровневой иерархии ролей для обеспечения большей гибкости при распределении прав доступа к системе.

СПИСОК ЛИТЕРАТУРЫ

1 Джон К. Митчелл, *Основания языков программирования*. НИЦ “Регулярная и хаотическая динамика”, 2010.

2 Benjamin Pierce, *Types and Programming Languages*. The MIT Press 2002.

3 Глазунов В.В., Кетов Д.В. Разработка системы управления учебными материалами на основе семантических моделей предметных областей //XXXIX НЕДЕЛЯ НАУКИ СПбГПУ.

4 Любченко В.В. Онтологическая модель знаний о предметных областях учебных курсов (стр. 269-275).

5 К вопросу о структуре учебного материала: [Электронный документ]. (http://www.dupliksv.hut.ru/pauk/papers/struct.html). Проверено 10.12.2010.

6 Краткое введение в Semantic Web: [Электронный документ]. (http://shcherbak.net/2008/01/brief_sw). Проверено 10.12.2010.

7 Модульная объектно-ориентированная динамическая учебная среда: [Электронный документ]. (http://moodle.org). Проверено 10.12.2010.

8 Описание формата LOM: [Электронный документ]. (http://www.elbib.ru/index.phtml?env_page=methodology/metadata/md_review/md_descrip_dl.html). Проверено 10.03.2011.

9 Основные виды и уровни организации учебно-познавательной деятельности учащихся: [Электронный документ]. (http://www.monographies.ru/76-2774). Проверено 10.03.2011.

10 Построение онтологии предметной области: [Электронный документ]. (http://masters.donntu.edu.ua/2010/fknt/bolotova/library/tez5.htm). Проверено 10.03.2011.

11 Применение web-онтологий в задачах дистанционного обучения: [Электронный документ]. (http://shcherbak.net/dist). Проверено 10.03.2011.

12 Принципы и методы создания курсов дистанционного обучения: [Электронный документ]. (http://www.ecsocman.edu.ru/univman/msg/145119.html). Проверено 10.03.2011.

13 Разработка онтологии дистанционного курса: [Электронный документ]. (http://shcherbak.net/webconf09-minsk-respublika-belorus/razrabotka-ontologii-distancionnogo-kursa). Проверено 10.03.2011.

14 Свободно распространяемые учебники и руководства: [Электронный документ]. (http://www.wikibooks.org). Проверено 10.12.2010.

15 Система управления курсами — Sakai: [Электронный документ]. (http://sakaiproject.org). Проверено 10.12.2010.

16 Сравнение тексто-ориентированных и логико-ориентированных Wiki: [Электронный документ]. (http://goranzugic.wordpress.com). Проверено 10.03.2011.

17 Структура учебной книги: [Электронный документ]. (http://www.ugatu.ac.ru/ddo/stuff/met_avtora/4.htm). Проверено 10.03.2011.

18 Терминология высшего профессионального образования: [Электронный документ]. (http://spbgpu.net). Проверено 10.03.2011.

19 Черновик стандарта LOM: [Электронный документ]. (http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf). Проверено 10.03.2011.

20 DjangoRDF Project: [Электронный документ]. (http://code.google.com/p/django-rdf) Проверено 10.03.2011.

21 GraphViz Java API: [Электронный документ]. (http://www.loria.fr/~szathmar/off/projects/java/GraphVizAPI/index.php). Проверено 04.05.2011.

22 Onto Wiki Features: [Электронный документ]. (http://ontowiki.net/Projects/OntoWiki/Features). Проверено 10.03.2011.

23 Ontopia API: [Электронный документ]. (http://www.ontopia.net/doc/current/doc/api) Проверено 30.04.2011. http://ontowiki.net/Projects/OntoWiki/Features

24 Ontopia Development Guide: [Электронный документ]. (http://www.ontopia.net/doc/current/doc/webed/devguide.html). Проверено 10.03.2011.

25 S. Pepper. The TAO of Topic Maps. Ontopia AS, 2001: [Электронный документ]. (http://www.ontopia.net/topicmaps/materials/tao.html). Проверено 10.12.2010.

26 Semantic MediaWiki Compare: [Электронный документ]. (http://smwforum.ontoprise.com/smwforum/index.php/Faq/Comparison_of_SMW%2B_with_other_semantic_wikis). Проверено 10.03.2011.

27 Semantic Web как новая модель информационного пространства Интернет: [Электронный документ]. (http://shcherbak.net/semantic-web-kak-novaya-model-informacionnogo-prostranstva). Проверено 10.12.2010.

28 Semantic Wiki Comparsion: [Электронный документ]. (http://imu.ntua.gr/Papers/C86-euroxxii_semantic%20wiki%20engines.pps). Проверено 10.03.2011.

29 The Core Concepts of CubicWeb: [Электронный документ]. (http://docs.cubicweb.org/intro/concepts.html). Проверено 10.12.2010.

30 TopicMaps Specification: [Электронный документ]. (http://www.topicmaps.org/xtm). Проверено 10.12.2010.

ПРИЛОЖЕНИЕ

```
<%@ page
import="
net.ontopia.topicmaps.nav2.core.*,
    net.ontopia.topicmaps.nav2.utils.FrameworkUtils,
    net.ontopia.topicmaps.query.utils.QueryUtils,
    java.util.Collection"
    contentType="text/html; charset=utf-8"
    %>
    <logic:context tmparam="tm" settm="topicmap">
    <%
    String tmid = request.getParameter("tm");
    String course = "topic-name($COURSE, $TN), " +
    "instance-of($COURSE, inst)?";
    %>
    <p align="center">
    Редактирование учебных материалов: <br /><b><logic:set name="
       course"><tm:tolog query="<%= query %>"select="COURSE"/>
    </logic:set></b><br />
    </p>
    <table style="width: 100%;">
    <tr>
    <td>
    <webed:field action="set-name" id="objName" type="short"
    params="OBJ">
    <tolog:out var="NAME"/>
    </webed:field>
    </td>
    <td> <webed:button action="submit" id="add". text='<%=
       pageContext.getAttribute("Add object").toString() %>'/> </td
       >
    </tr>
    <tr> <td> <%
    <------>Collection<String> links = QueryUtils.getAvailableLinks
       ();
    <------>for (String link : links) {
        <------>  StringBuilder sb = new StringBuilder(language);
        <------>  String desc = sb.toString();
```

```
%><option value="<%= links %>"><%= desc %></option>

    </td>
    <td>
    <input type="submit" value="Добавить связь" id="
       add_link" />
    <logic:foreach name="sourceLocators">[<output:locator
       />]
    </logic:foreach>
    <output:objectid of="assoc"/>
    </logic:else>
    </logic:if>

    <logic:foreach name="objects">
    <logic:set name="object" comparator="off"><tm:topics
       /></logic:set>
    <logic:set name="link" comparator="off"><tm:classesOf
       /></logic:set>
    <tr><td>
    <logic:if name="object"><logic:then>
    <output:element name="a">
    <output:attribute name="href"><output:link of="link"
    template="course_edit.jsp?tm=%topicmap%&amp;id=%id%"
    generator="modelLinkGenerator"/></output:attribute>
    <output:name of="object"/>
    </output:element></logic:then></logic:if></td>
    <td><output:element name="a"><output:attribute name="
       href"><output:link of="type"
    template="course_view.jsp?tm=%topicmap%&amp;id=%id%"
    generator="modelLinkGenerator"/></output:attribute><
       output:name of="type"/></output:element> </td> </tr>
    </table>
    <p align="center">
    Список объектов онтологии:</p>
    <table style="width:100%;">
    <tr>
    <td align="right">
    Фильтр класса:</td>
    <td>
```

```
    <tolog:foreach query="role-filter($R1, %course%), type(
        $R1, after),
association-role($ASSOC, $R1),
association-role($ASSOC, $R2), type($R2, prior),
role-filter($R2, $OTHER)?">

    <webed:list action="assign-filter"
    params="ASSOC t:depends-on class t:after t:prior"
    collection="classes"
    selected="OTHER"/> <br>
    </tolog:foreach>

    <webed:list action="assign-object"
    params="empty t:depends-on material t:after t:prior"
    collection="courses"/> <br>
    </td>
    </tr>
    </table>
    <table style="width: 100%;">
    <tr>
    <td>
    Имя объекта:</td>
    <td>
     </td>
    <td>
     </td>
    <td>
     </td>
    <td>
     </td>
    <td>
    Имя класса:</td>
    </tr>
    <%
    ContextUtils.setValue("types", pageContext, types);
%>
    <logic:foreach name="types" set="type" comparator="
        topicComparator">
    <%
```

```
        TopicIF topic = (TopicIF)ContextUtils.getSingleValue("
            type", pageContext);
      Collection tmp = tindex.getAssociations(topic);
      ContextUtils.setValue("assoctypes", pageContext, tmp);
      %>
          <tr><td>
          <a href="
          <---><------> <output:link of="topicmap"
          <---><------><------>       template="view_obj.jsp?tm=%
topicmap%&id="
          <---><------><------>       generator="
modelLinkGenerator"/><output:objectid of="type"/>">
          <---><------> <output:name of="type"/><img src="img/
              view.jpeg" />
          <---><------></a>
          </td>
          <td><a href="
          <---><------> <output:link of="topicmap"
          <---><------><------>       template="edit_obj.jsp?tm=%
topicmap%&id="
          <---><------><------>       generator="
modelLinkGenerator"/><output:objectid of="type"/>">
          <---><------> <output:name of="type"/><img src="img/
              edit.jpeg" />
          <---><------></a>
          </td>
          <td><a href="
          <---><------> <output:link of="topicmap"
          <---><------><------>       template="edit_links.jsp?tm
=%topicmap%&id="
          <---><------><------>       generator="
modelLinkGenerator"/><output:objectid of="type"/>">
          <---><------> <output:name of="type"/><img src="img/
              edit-properties-icon-small.gif" />
          <---><------></a>
          </td>
          <td><a href="
          <---><------> <output:link of="topicmap"
          <---><------><------>       template="remove_obj.jsp?tm
=%topicmap%&id="
```

```
        <---><------><------>        generator="
modelLinkGenerator"/><output:objectid of="type"/>">
        <---><------> <output:name of="type"/><img src="img/
            remove.jpeg" />
        <---><------></a>
        </td>
        <td><tolog:foreach query="role-filter($R1, %material%),
            type($R1, after),
      association-role($ASSOC, $R1),
      association-role($ASSOC, $R2), type($R2, prior),
      role-filter($R2, $OTHER)?">
        </td>
        </tr>
        </logic:foreach>
        </table>
        <div>
        <input type="hidden" value="<%= tmid %>" name=tm />
        </div></form>
        </logic:context>
        <logic:context tmparam="tm" settm="topicmap">
        <%
        String tmid = request.getParameter("tm");
    %>
        <div>
        <table style="width:100%;">
        <tr>
        <td>
        Курс:</td>
        <b>
        <td>
        <tolog:foreach query="select $ASSOC, $OTHER from
        role-material($R1, %course%), type($R1, after),
      association-role($ASSOC, $R1), type($ASSOC, depends-on),
      association-role($ASSOC, $R2), type($R2, prior),
      role-material($R2, $OTHER)?">
        <webed:list action="assign-course"
        params="ASSOC t:depends-on course t:after t:prior"
        collection="courses"
        selected="OTHER"/> <br>
        </tolog:foreach>
```

```
    </td>
    <td>
    </webed:form>
    </tr>
    </table>
    </b>
    <br />
    </div>
    <table style="width:100%;">
    <tr>
    <td>
    Элемент структуры курса:</td>
    <td>
    <tolog:set var="material-assoc"
    query="select $ASSOC from
    role-object($R1, %course%), type($R1, course),
association-role($ASSOC, $R1), type($ASSOC, has-course),
association-role($ASSOC, $R2), type($R2, status),
role-object($R2, $OTHER)?"/>
    <tolog:set var="material"
    query="has-course(%material% : course, $MATERIAL :
       material)?"/>
    <webed:list action="assign-material"
    params=" t:material-assoc t:has-material course t:
       course"
    collection="materials"/>
    </td>
    </tr>
    </table>
    <br />
    Добавление материала курса:<br />
    <br />
    <table style="width:100%;">
    <tolog:foreach query="role-material($R1, %course%),
       type($R1, after),
association-role($ASSOC, $R1),
association-role($ASSOC, $R2), type($R2, prior),
role-material($R2, $OTHER)?">
    <webed:list action="assign-material"
    params="ASSOC t:depends-on material t:after t:prior"
```

```
collection="materials"
selected="OTHER"/> <br>
</tolog:foreach>
<webed:list action="assign-material"
params="empty t:depends-on material t:after t:prior"
collection="materials"/> <br>
</table>
<br />
Значение:<br />
<textarea name="query" rows="15" cols="62" tabindex="2"
  ><%= (request.getParameter("value") == null ? "" :
   request.
        getParameter("value")) %></textarea>
<br />
<br />
<input id="Добавить" type="button" value="Добавить" /><
   br />
<br />
Список зависимых понятий:<br />
<br />
<table style="width: 100%;">
<tr>
<td>
Имя объекта:</td>
<td>
Отображение:</td>
<td>
 </td>
</tr>
<tr><td>
<tolog:set var="hasObject" query="select $HAS-OBJECT
   from type($HAS-OBJECT, has-object), association-role
   ($HAS-OBJECT, $OBJECT), role-object($OBJECT, %OBJECT
   %)?"/>
</td>
<td>
<webed:checkbox action="assign" id="checkHasObject"
params="hasObject"/>
</td>
</table>
```

```
    <p>
    <webed:button action="view" text="Сохранить отображение
       "/>
    </logic:context>
    <%@ page import="net.ontopia.topicmaps.core.*, net.
       ontopia.topicmaps.entry.*,
    net.ontopia.topicmaps.utils.*,
net.ontopia.topicmaps.nav2.utils.*"%><%
    TopicMapRepositoryIF rep = ContextUtils.getRepository(
       pageContext.getServletContext());
%>
    <tolog:context topicmap="course.xtm">
    <tolog:set var="tm" reqparam="id"/>
    <tolog:declare>
    using t for s"#"
    </tolog:declare>
    <webed:form actiongroup="course">
    <p>
    <b>Курс</b>: <logic:set name="course"><tm:tolog query="
       <%= query %>" select="COURSE"/></p>
    <tolog:foreach query="object($R1, %courses%), type($R1,
        after),
object($ASSOC, $R1),
object($ASSOC, $R2), type($R2, prior),
object($R2, $OTHER)?">
    <webed:list action="assign-course"
    params="ASSOC t:depends-on course t:after t:prior"
    collection="courses"
    selected="OTHER"/> <br>
    <tolog:set var="name" query="topic-name(%course%, $TN)?
       "/>
    <tolog:set var="desc"
    query="occurrence(%course%, $OCC), type($OCC, desc)?"/>
    <tolog:set var="object" query="instance-of($SUBJECT,
       subject)?"/>
    <tolog:set var="subject"
    query="assigned-to(%course% : assignment, $P :
       responsible)?"/>
    <tolog:set var="assign-assoc"
```

```
            query="select $A from
            tm($R, %course%),
        object($A, $R),
        type($A, assigned-to)?"/>
            </tolog:foreach>
            </webed:form>

            <%@ page import="net.ontopia.topicmaps.core.*, net.
                ontopia.topicmaps.entry.*,
    net.ontopia.topicmaps.utils.*,
net.ontopia.topicmaps.nav2.utils.*,java.io.File;"%><%
    TopicMapRepositoryIF rep = ContextUtils.getRepository(
        pageContext.getServletContext());
OntViz = new OntViz();
OntViz gv = new OntViz();
......
    String type = "png";
File out = new File("img/out." + type);
gv.writeGraphToFile( gv.getGraph( gv.getDotSource(), type ), out );
%>
    <p align="center">
    Просмотр онтологии учебных материалов: <br /><b><logic:set name
        ="course"><tm:tolog query="<%= query %>"select="COURSE"/></b
        ></p>
    <div>
    <img src="img/out.png" style="border-width:0px;" />
    </div>

import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.InputStreamReader;
public class OntViz
{
    private static String TEMP_DIR = "/tmp";
    private static String DOT = "/usr/bin/dot";
    private StringBuilder graph = new StringBuilder();
```

```
public OntViz() {
}
public String getDotSource() {
    return graph.toString();
}
public void add(String line) {
    graph.append(line);
}
public void addln(String line) {
    graph.append(line + "\n");
}
public void addln() {
    graph.append('\n');
}
public byte[] getGraph(String dot_source, String type)
{
    File dot;
    byte[] img_stream = null;

    try {
        dot = writeDotSourceToFile(dot_source);
        if (dot != null)
        {
            img_stream = get_img_stream(dot, type);
            return img_stream;
        }
        return null;
    } catch (java.io.IOException ioe) { return null; }
}
public int writeGraphToFile(byte[] img, String file)
{
    File to = new File(file);
    return writeGraphToFile(img, to);
}
public int writeGraphToFile(byte[] img, File to)
{
    try {
        FileOutputStream fos = new FileOutputStream(to);
        fos.write(img);
        fos.close();
```

```
        } catch (java.io.IOException ioe) { return -1; }
        return 1;
    }
    private byte[] get_img_stream(File dot, String type)
    {
        File img;
        byte[] img_stream = null;

        img = File.createTempFile("graph_", "."+type, new File(
            OntViz.TEMP_DIR));
        Runtime rt = Runtime.getRuntime();
        String[] args = {DOT, "-T"+type, dot.getAbsolutePath(), "-o
            ", img.getAbsolutePath()};
        Process p = rt.exec(args);
        p.waitFor();
        FileInputStream in = new FileInputStream(img.
            getAbsolutePath());
        img_stream = new byte[in.available()];
        in.read(img_stream);
        if( in != null ) in.close();
    }
    return img_stream;
}
private File writeDotSourceToFile(String str) throws java.io.
    IOException
{
    File temp;
    temp = File.createTempFile("graph_", ".dot.tmp", new File(
        OntViz.TEMP_DIR));
    FileWriter fout = new FileWriter(temp);
    fout.write(str);
    fout.close();
}
return temp;
    }
public String start_graph() {
    return "digraph G {";
}
public String end_graph() {
    return "}";
```

```
}
public void readSource(String input)
{
    StringBuilder sb = new StringBuilder();
    FileInputStream fis = new FileInputStream(input);
    DataInputStream dis = new DataInputStream(fis);
    BufferedReader br = new BufferedReader(new InputStreamReader(
       dis));
    String line;
    while ((line = br.readLine()) != null) {
        sb.append(line);
    }
    dis.close();
    this.graph = sb;
}

}

import java.io.File;
import javax.servlet.ServletContext;
import javax.servlet.http.HttpServletRequest;
import net.ontopia.topicmaps.core.TopicMapIF;
import net.ontopia.topicmaps.nav2.plugins.DefaultPlugin;
import net.ontopia.topicmaps.nav2.plugins.PluginIF;
import net.ontopia.topicmaps.nav2.taglibs.logic.ContextTag;
import net.ontopia.utils.OntopiaRuntimeException;
public class Fulltext extends DefaultPlugin {
public String generateHTML(ContextTag context) {
    ServletContext ctxt = context.getPageContext().
       getServletContext();
    HttpServletRequest request =
        (HttpServletRequest)context.getPageContext().getRequest();
    String tm = context.getTopicMapId();
    boolean exists;
    String path = ctxt.getRealPath("/course/" + tm);
    TopicMapIF topicmap = context.getTopicMap();
    if (topicmap != null && topicmap.getStore().getProperty("net.
       ontopia.infoset.fulltext.impl.rdbms.RDBMSSearcher.type") !=
       null) {
        exists = true;
```

```
        } else {
            exists = new File(path).exists();
        }
        if (!exists) {
            PluginIF admin_plugin = context.getNavigatorConfiguration()
               .getPlugin("fulltext");
        }
        String action = request.getContextPath() + "/" + getURI();
        String query = getParameter("query");
        if (query == null) query = "";
        String query_size = getParameter("query-size");
        if (query_size == null) query_size = "10";
        String type = getParameter("type");
        if (type == null || type.equals("form")) {
            StringBuffer sb = new StringBuffer();
            sb.append("<form action='").append(action)
                .append("' method='get' style='display: inline'");
            if (description != null)
                sb.append(" title=\"").append(description).append('\"')
                   ;
            sb.append('>')
                .append("<input type='hidden' value='").append(tm).
                   append("' name='tm'>")
                .append("<input type='text' name='query' size='").
                   append(query_size)
                .append("' value='").append(query).append("'>")
                .append("</form>");
            return sb.toString();
        } else {
            return super.generateHTML(context);
        }
    }
}

import java.io.IOException;
import java.util.Collection;
import javax.servlet.ServletRequest;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.TagSupport;
```

```
import net.ontopia.topicmaps.core.*;
import net.ontopia.topicmaps.query.core.*;
import net.ontopia.topicmaps.query.utils.QueryUtils;
import net.ontopia.topicmaps.nav2.core.FunctionIF;
import net.ontopia.topicmaps.nav2.core.NavigatorRuntimeException;
import net.ontopia.topicmaps.nav2.impl.basic.JSPEngineWrapper;
import net.ontopia.topicmaps.nav2.impl.basic.AbstractFunction;
import net.ontopia.topicmaps.nav2.utils.ContextUtils;
import net.ontopia.topicmaps.nav2.utils.TreeWidget;


public class TopicMapQuery extends AbstractFunction {
    public Collection execute(PageContext pageContext, TagSupport
       callingTag)
        throws IOException, JspException {
            ServletRequest request = pageContext.getRequest();
            JspWriter out = pageContext.getOut();
            String id = request.getParameter("id");
            String tmid = request.getParameter("tm");
            TopicIF atype = (TopicIF) ContextUtils.getSingleValue("
               topic", pageContext);
            TopicMapIF topicmap = atype.getTopicMap();
            QueryProcessorIF processor = QueryUtils.
               getQueryProcessor(topicmap);
            String a = atype.getObjectId();
            String p = parent.getObjectId();
            String c = child.getObjectId();
            String topquery =
                "select $OBJECT from " +
                "  @" + a + "($OBJECT : @" + p + ", $CHILD : @" + c
                    + "), " +
                "  not(@" + a + "($OBJECT : @" + c + ", $PARENT : @
                   " + p + ")) " +
                "order by $OBJECT?";
            String query =
                "  @" + a + "(%parent% : @" + p + ", $CHILD : @" +
                   c + ") " +
                "order by $CHILD?";
            TreeWidget widget = new TreeWidget(topicmap, query,
               topquery,
```

```
                    "?id=" + id + "&tm=" + tmid + "&",
                    "topic_complete.jsp?tm=" + tmid + "&");
            widget.setWidgetName("/course/" + tmid);
            widget.setImageUrl("/course/img/");
            widget.run(pageContext, out);
        }
    private TopicIF query(QueryProcessorIF processor, String query)
        throws InvalidQueryException {
            QueryResultIF result = null;
            try {
                result = processor.execute(query);
                if (result.next())
                    return (TopicIF) result.getValue(0);
                else
                    return null;
            } finally {
                if (result != null)
                    result.close();
            }
        }
}


import java.io.IOException;
import java.util.Arrays;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import javax.servlet.jsp.JspWriter;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.JspTagException;
import javax.servlet.jsp.PageContext;
import javax.servlet.jsp.tagext.BodyTagSupport;
import javax.servlet.jsp.tagext.BodyContent;
import net.ontopia.topicmaps.utils.TopicComparators;
import net.ontopia.topicmaps.utils.TopicStringifiers;
import net.ontopia.utils.StringifierIF;
import net.ontopia.topicmaps.nav2.utils.NavigatorUtils;
import net.ontopia.topicmaps.nav2.core.*;
import net.ontopia.topicmaps.nav2.utils.FrameworkUtils;
```

```
import org.apache.commons.collections.comparators.ReverseComparator
    ;


public class ForEachTag extends BodyTagSupport {
private static final StringifierIF DEF_TOPIC_STRINGIFIER =
    TopicStringifiers
     .getDefaultStringifier();
private static final Comparator DEF_TOPIC_COMPARATOR =
    TopicComparators
     .getCaseInsensitiveComparator(DEF_TOPIC_STRINGIFIER);
private ContextManagerIF ctxtMgr;
private ContextTag contextTag;
private Object[] items;
private int index;
private String      collVariableName;
private String      itemVariableName;
private Comparator listComparator;
private String      listComparatorClassName;
private int         maxNumber;
private int         startNumber;
private String      separator;
private boolean     sortItemsFlag;
private String      sortOrder;
private String      functionOnTruncate;
public ForEachTag() {
    super();
    initializeValues();
}
public int doStartTag() throws JspTagException {
this.contextTag = FrameworkUtils.getContextTag(pageContext);
this.ctxtMgr = contextTag.getContextManager();
Collection coll = null;
if (collVariableName != null)
    coll = ctxtMgr.getValue(collVariableName);
else
    coll = ctxtMgr.getDefaultValue();
if (maxNumber <= 0)
    maxNumber = contextTag.getNavigatorConfiguration()
        .getProperty(NavigatorConfigurationIF.MAX_LIST_LENGTH,
```

```
                    NavigatorConfigurationIF.DEF_VAL_MAX_LIST_LENGTH);
ctxtMgr.pushScope();
if (coll.isEmpty())
    return SKIP_BODY;
this.items = coll.toArray();
if (sortItemsFlag) {
    try {
        listComparator = getComparator();
        if (listComparator == null)
            listComparator = DEF_TOPIC_COMPARATOR;
        Arrays.sort( items, listComparator);
    }
    if (startNumber >= items.length)
        startNumber = items.length - 1;
    index = startNumber;
    setVariableValues(items[index]);
    index = startNumber + 1;
    return EVAL_BODY_BUFFERED;
}
public int doAfterBody() throws JspTagException {
    BodyContent body = getBodyContent();
    JspWriter out = body.getEnclosingWriter();
    out.print(body.getString());
    body.clearBody();
    if (index < items.length
            && index < maxNumber) {
        if (separator != null) {
            out.print( separator );
        }
        setVariableValues(items[index]);
        index++;
        return EVAL_BODY_AGAIN;
    } else {
        return SKIP_BODY;
    }
}
public int doEndTag() throws JspException {
    if (index >= maxNumber) {
        String functionName = null;
        if (functionOnTruncate != null)
```

```
                functionName = functionOnTruncate;
            else
                functionName = contextTag.getNavigatorConfiguration()
                    .getProperty(NavigatorConfigurationIF.
                        DEF_FUNC_ONTRUNCATE);
        }
        ctxtMgr.popScope();
        ctxtMgr = null;
        contextTag = null;
        items = null;
        return EVAL_PAGE;
    }
    public void setName(String collectionName) {
        this.collVariableName = collectionName;
    }
    public void setSet(String itemName) {
        this.itemVariableName = itemName;
    }
    public void setComparator(String classname) {
        this.listComparatorClassName = classname;
        this.sortItemsFlag = (classname != null);
    }
    public void setOrder(String order) throws NavigatorRuntimeException
        {
        this.sortOrder = order;
        this.sortItemsFlag = (order != null);
    }
    public void setMax(String maxString) {
        try {
            this.maxNumber = Integer.parseInt( maxString );
        } catch (NumberFormatException e) {
            this.maxNumber = 0;
        }
    }
    public void setStart(String startString) {
        try {
            this.startNumber = Integer.parseInt( startString );
        } catch (NumberFormatException e) {
            this.startNumber = 0;
        }
```

```
        if (this.startNumber < 0)
            this.startNumber = 0;
    }
    public void setSeparator(String separator) {
        this.separator = separator;
    }
    public void setFunctionOnTruncate(String functionName) {
        this.functionOnTruncate = functionName;
    }
    private void setVariableValues(Object elem) {
        ctxtMgr.setDefaultValue( elem );
        if (itemVariableName != null)
            ctxtMgr.setValue(itemVariableName, elem);
        ctxtMgr.setValue(NavigatorApplicationIF.FOREACH_SEQ_INDEX_KEY,
                new Integer(index+1));
        if (index == startNumber)
            ctxtMgr.setValue(NavigatorApplicationIF.
               FOREACH_SEQ_FIRST_KEY,
                     Boolean.TRUE);
        else
            ctxtMgr.setValue(NavigatorApplicationIF.
               FOREACH_SEQ_FIRST_KEY,
                     Collections.EMPTY_LIST);
        if (index < items.length-1
                && index < maxNumber-1)
            ctxtMgr.setValue(NavigatorApplicationIF.
               FOREACH_SEQ_LAST_KEY,
                     Collections.EMPTY_LIST);
        else
            ctxtMgr.setValue(NavigatorApplicationIF.
               FOREACH_SEQ_LAST_KEY,
                     Boolean.TRUE);
    }
    private Comparator getComparator() throws NavigatorRuntimeException
        {
        Object obj = contextTag.getNavigatorApplication()
            .getInstanceOf(listComparatorClassName);
        if (obj != null && obj instanceof Comparator)
            return (Comparator) obj;
        else
```

```
        return null;
}
private void initializeValues() {
    collVariableName = null;
    itemVariableName = null;
    listComparatorClassName = null;
    sortItemsFlag = false;
    sortOrder = null;
    maxNumber = -1;
    startNumber = 0;
    separator = null;
    functionOnTruncate = null;
}
}
```

ЗАКЛЮЧИТЕЛЬНЫЙ ЛИСТ РАБОТЫ

Магистерская диссертация выполнена мною самостоятельно. Использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.
Список использованной литературы (источников) 30 наименований.
Работа выполнена на 63 листах.
Приложения к работе на 20 листах.

Один экземпляр сдан на кафедру.

Подпись _____/_____/

(фамилия, инициалы)

Дата «_____»_____ 20_____г.